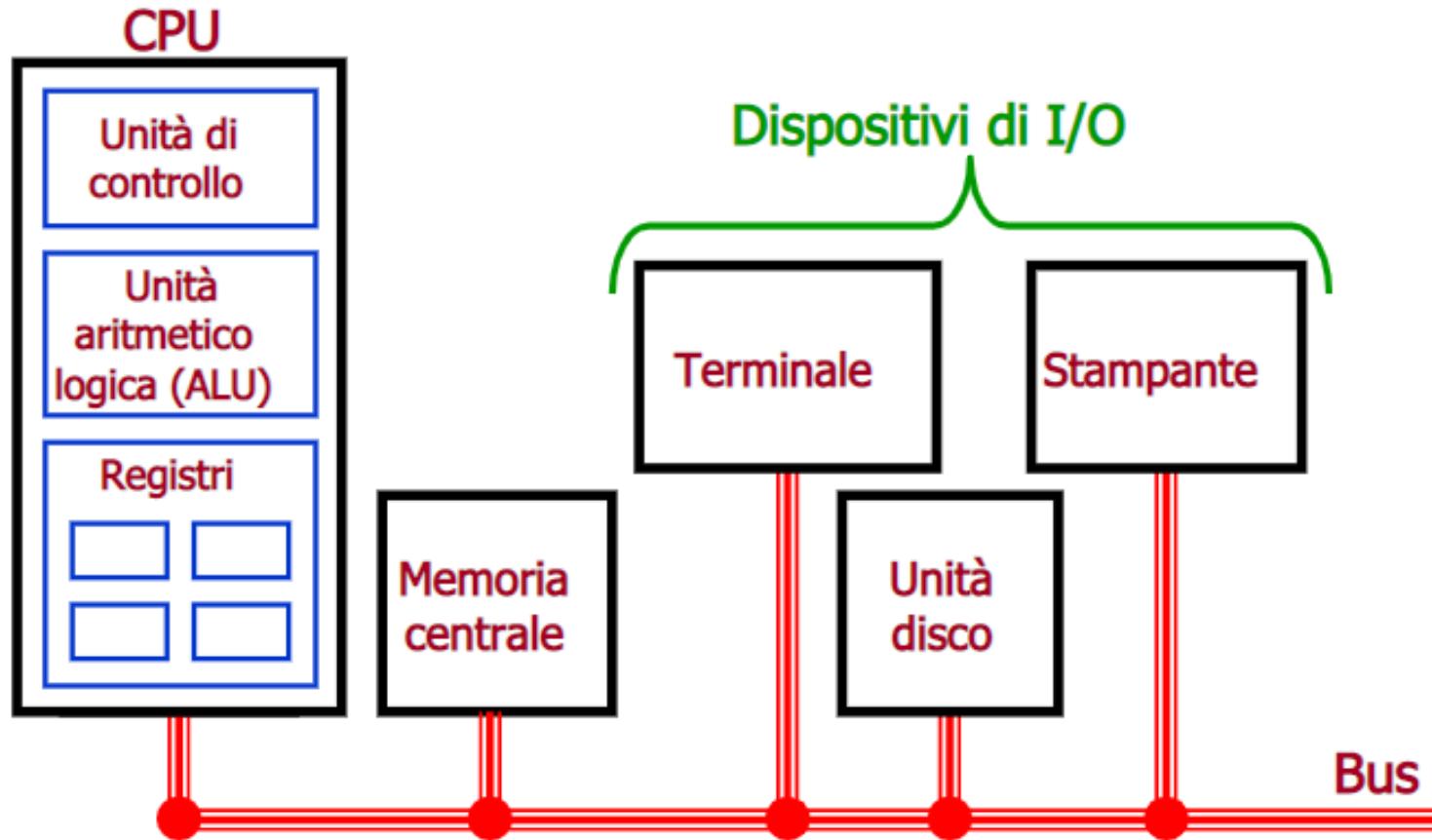


# Struttura del Calcolatore



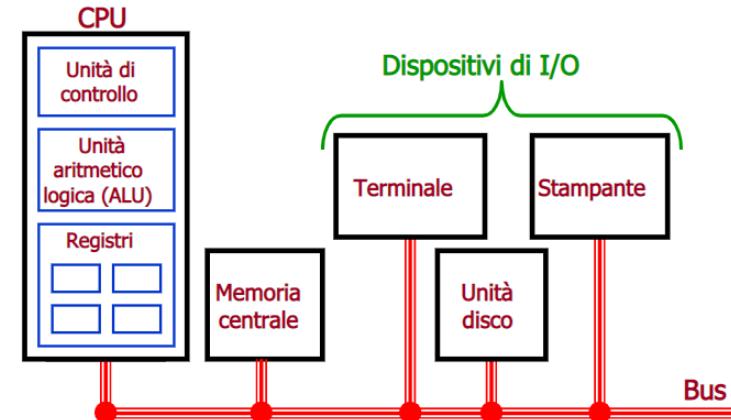
Corso di Architettura degli Elaboratori 1  
Fulvio Valenza  
A.A. 2018-2019  
Polo di Vercelli

# Struttura di un semplice computer



# Struttura di un semplice computer

- La **CPU** Central Processing Unit è il “cervello” del computer e la sua funzione è quella di eseguire i programmi contenuti nella **Memoria Centrale**.
  - La CPU è composta a: **Control Unit**, **ALU** (Arithmetic Control Unit) e dai **Registri**.
- I componenti sono connessi fra loro mediante un (o più) **Bus**, ovvero una serie di cavi paralleli sui quali vengono trasmessi, indirizzi, dati e segnali di controllo
- Dispositivi di Input e Output

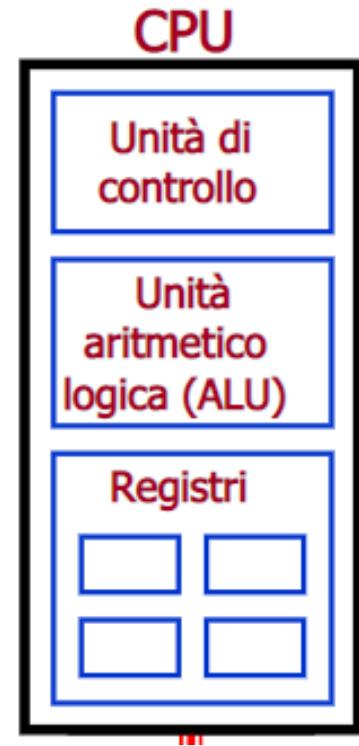


# CPU

# Organizzazione della CPU

---

- **Control Unit:** si occupa di coordinare tutte le azioni necessarie per l'esecuzione di una/più **istruzione**, ovvero prelevare le istruzioni dalla memoria principale e determinarne il tipo.
- **ALU:** esegue le operazioni aritmetico-logiche necessarie per portare a termine l'esecuzione delle istruzioni.



# Organizzazione della CPU

- **Registri:** piccola quantità di memoria ad alta velocità (essendo interni alla CPU), utilizzata per memorizzare i risultati temporanei di alcune operazioni e alcune informazioni di controllo.

Ogni registro:

- ha una funzione e una dimensione predefinite
- può contenere un numero, il cui valore può variare fino a un massimo dipendente dalla dimensione del registro.



# Registri principali

---

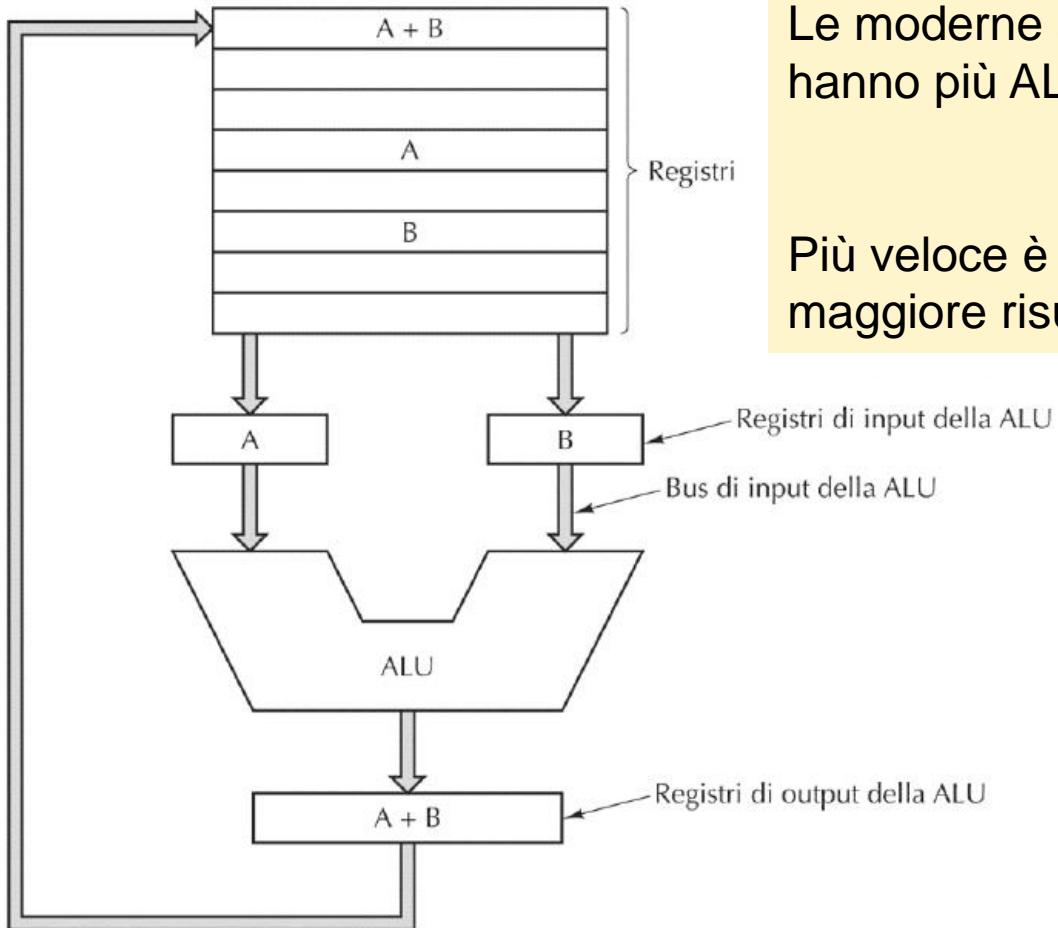
- **Program counter (PC):** ha lo scopo di tenere traccia (puntare) alla successiva istruzione che dovrà essere prelevata per l'esecuzione.
- **Instruction Register IR:** contiene l'istruzione che si trova in fase di esecuzione

# Tipi di istruzioni

---

- La maggior parte di istruzioni può essere divisa in due categorie principali: **registro-memoria** e **registro-registro**
- Registro-Memoria: permettono di prelevare parole dalla memoria per portarle all'interno dei registri o copiare i valori dei registri alla memoria
- Registro-Registro: un esempio di operazioni di questo tipo, consiste nel prelevare due operandi all'interno di due registri e portarli come input alla ALU, il quale output verrà memorizzato in un altro registro.

# Percorso dati



Le moderne architetture dei calcolatori hanno più ALU che lavorano in parallelo.



Più veloce è il ciclo del percorso dati, maggiore risulta la velocità del calcolatore

**Figura 2.2** Percorso dati di una tipica macchina di von Neumann.

# Ciclo delle istruzioni: FETCH-DECODE-EXECUTE

---

La CPU esegue ogni istruzione compiendo una serie di piccolo passi, che in linea generale sono:

- 
1. Prelevare la prossima istruzione dalla memoria (indirizzo in PC) e inserirla nel IR FETCH
  2. Aggiornare il PC
  3. Determinare l'istruzione appena caricata DECODE
  4. Se l'istruzione usa dati in memoria individua dove si trovano (i loro indirizzi) e li preleva
  5. Eseguire l'istruzione EXECUTE

# Linguaggio Macchina

---

- Ciascun calcolatore ha un ampio insieme di istruzioni che è in grado di eseguire, rappresentate mediante sequenze di bit
- Poiché il numero di istruzioni in un programma possono essere anche migliaia è assai facile commettere errori
- Scrivere un programma in linguaggio macchina richiede molto tempo.

# Linguaggi ad alto livello

---

- Le stesse operazioni in linguaggio macchina possono essere descritte sotto forma di un programma descritto con un linguaggio di alto livello
  
- I linguaggi di alto livello oltre a semplificare la definizione di codice, permettono di *Mantenere più semplice l'architettura hardware* sottostante e di estendere più facilmente l'insieme di istruzioni messe a disposizione dal processore.

# Traduzione dei linguaggi ad alto livello

---

- I linguaggi di alto livello richiedono una traduzione in linguaggio macchina per essere eseguiti.
  
- Tale attività può essere realizzata tramite:
  - compilazione
  - interpretazione;

# Traduzione dei linguaggi ad alto livello

---

## Compilatore

- ❑ Il programma compilatore analizza e traduce nella sua interezza il programma sorgente.
- ❑ Il risultato della compilazione è un programma oggetto.
- ❑ Solo dopo la traduzione dell'intero programma è possibile eseguirlo
- ❑ Vantaggi del compilatore:
  - ❑ migliori prestazioni nell'esecuzione;
  - ❑ il codice ottenibile può essere ottimizzato
  - ❑ La compilazione porta ad applicativi più veloci rispetto all'interpretazione a discapito del tempo di sviluppo.

# Traduzione dei linguaggi ad alto livello

---

## Interprete

- ❑ Il programma interprete analizza e traduce istruzione per istruzione in linguaggio macchina.
- ❑ Non appena una istruzione è interpretata può essere eseguita.
- ❑ Vantaggi dell'interprete:
  - ❑ consente una minore occupazione di memoria;
  - ❑ minore costo nella modifica dei programmi.
  - ❑ L'interprete consente tempi di sviluppo minori a discapito dell'efficienza nell'esecuzione.

# Microinterprete

---

- Il microinterprete viene memorizzato in una CONTROL STORE: una memoria di sola lettura (ROM) molto veloce, per ridurre il ritardo introdotto dall'interprete rispetto all'esecuzione diretta.
- Il confine fra hardware e software non è così netto.

# RISC v.s. CISC

---

- Durante i tardi anni '70 gli interpreti permisero di sperimentare istruzioni molto complesse;
  - il tentativo inseguito dai progettisti era quello di ridurre il “gap semantico” che divideva ciò che erano in grado di fare le macchine da quello che invece richiedevano i linguaggi di programmazione ad alto livello.
- All'inizio degli anni '80, alcuni ricercatori andarono contro corrente cercando di costruire processori in grado di eseguire molto più rapidamente le istruzioni più frequenti, non utilizzando alcun interprete.

# RISC v.s. CISC

---

- Si creò quindi una guerra di ‘religione’ fra i sostenitori delle soluzione CISC (VAX, Intel e i grandi mainframe IBM) e i sostenitori della soluzione RISC
  
- L’acronimo RISC significa infatti computer con un insieme ridotto d’istruzioni (Reduced Instruction Set Computer), scelto in contrapposizione a CISC, che significa computer con un insieme d’istruzioni complesso (Complex Instruction Set Computer)

# RISC v.s. CISC

---

- I sostenitori del soluzione CISC sostenevano che il miglior modo per progettare un calcolatore fosse quello di avere poche istruzioni
  
- La loro tesi era che una macchina RISC prevaleva nei confronti di una CISC perché, anche se impiegava quattro o cinque istruzioni per fare ciò che una macchina CISC realizzava con una sola, le sue istruzioni erano 10 volte più veloci (in quanto non interpretate).

# RISC v.s. CISC

---

- Si potrebbe pensare che, grazie ai vantaggi offerti dalla tecnologia RISC, queste macchine (come la Sun UltraSPARC) abbiano fatto scomparire dal mercato le macchine CISC (come gli Intel Pentium), e invece d ciò non è avvenuto:
  - Primo fra tutti va considerato il problema della retrocompatibilità e con esso i miliardi di dollari investiti dalle società per produrre il software per le macchine Intel.
  - Secondo, Intel è riuscita sorprendentemente a impiegare le stesse idee anche all'interno della architettura CISC.

# RISC v.s. CISC

---

- A partire dal processore 486, le CPU di Intel contengono un nucleo di tipo RISC che esegue le istruzioni più semplici (e spesso più comuni) in un unico ciclo dei percorsi dati, mentre interpreta le istruzioni più complesse secondo la classica modalità CISC.
- Il risultato finale è che le istruzioni più comuni sono veloci, mentre quelle meno comuni sono più lente.
- Anche se questo approccio non è altrettanto veloce quanto una pura architettura RISC, esso fornisce prestazioni generali competitive, permettendo allo stesso tempo di eseguire senza alcuna modifica il software progettato precedentemente.

# Parallelismo

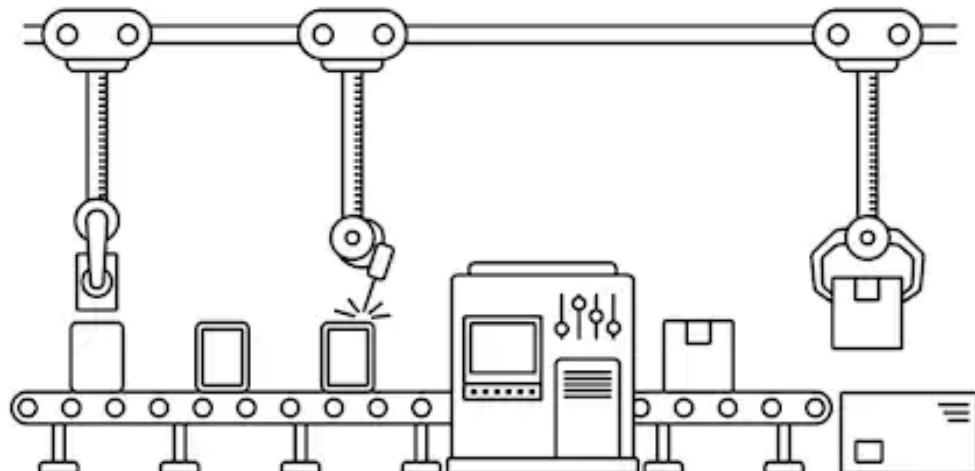
---

- I progettisti di calcolatori si sforzano costantemente di migliorare le prestazioni delle loro macchine.
- Aumentare la velocità di clock per rendere i processori più veloci è una possibilità, ma per qualsiasi nuova architettura esiste un limite, dipendente dal momento storico, su che cosa sia possibile ottenere mediante la semplice forza bruta.
- Per questo morivo molti progettisti di computer vedono nel parallelismo (cioè nel compiere più azioni allo stesso tempo) un modo per ottenere prestazioni più elevate con una data velocità di clock.

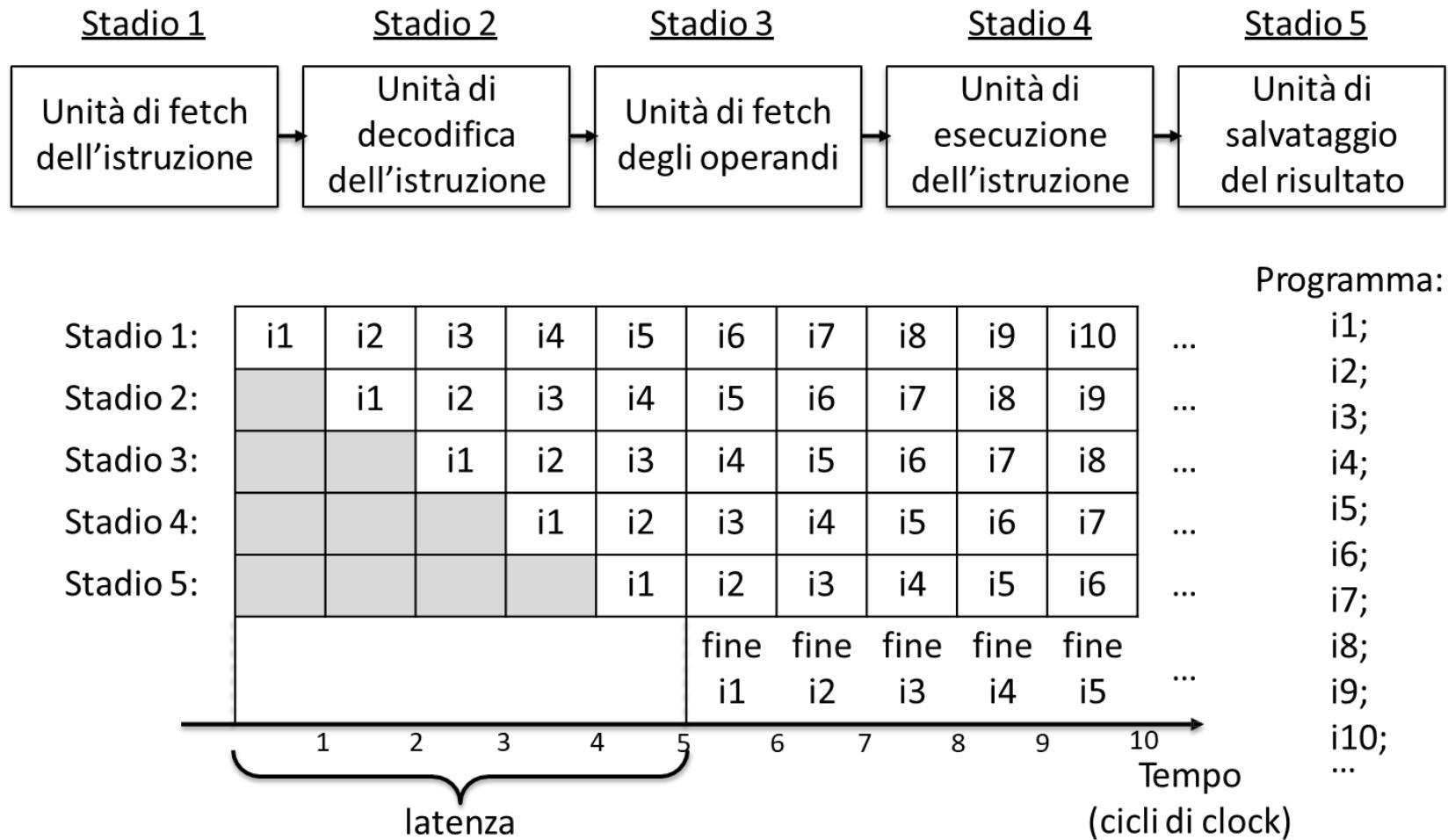
# Pipeline

---

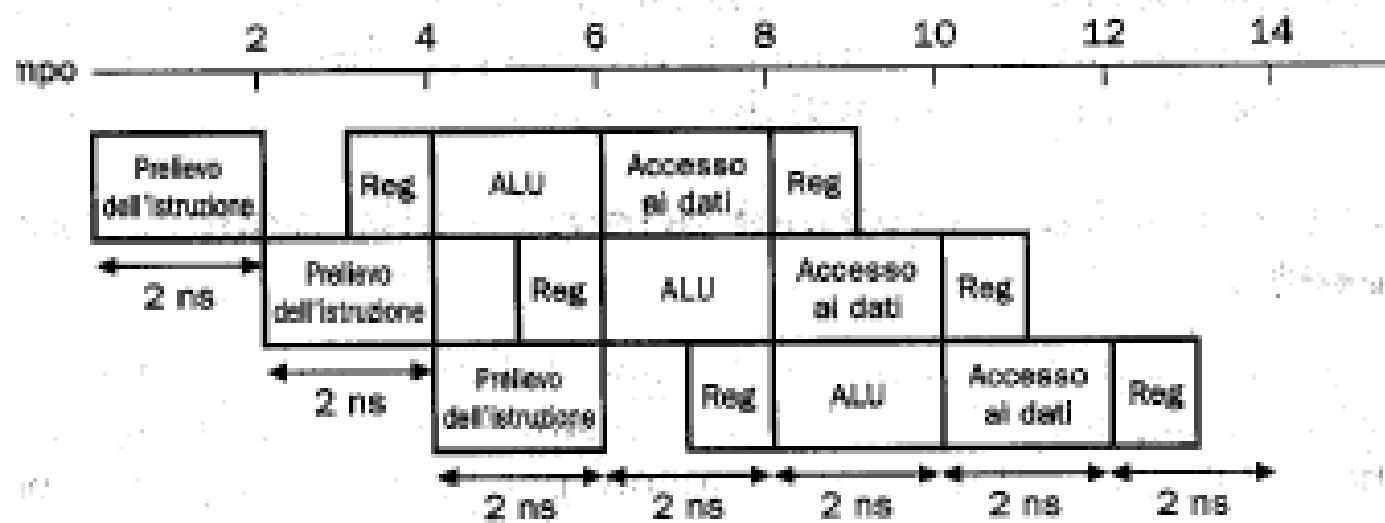
- Se per costruire un oggetto servono 4 operazioni, ciascuna delle quali impiega 1 secondo, il tempo totale per completare un oggetto è di 4 secondi, ma se ciascuna fase può essere svolta da una entità diversa, si può far entrare nel sistema 1 nuovo pezzo ogni secondo, e quindi uscirà dall'impianto 1 oggetto al secondo, anziché 1 oggetto ogni 4 secondi.



# Pipeline Ideale



# Pipeline imperfetta o reale



# Pipeline

---

NON CONFONDERE:

- ❑ Tempo di esecuzione di una istruzione ( $T_i$ ) che è dato dalla somma dei tempi necessari a percorrere tutti gli stadi di pipeline
  
- ❑ Numero di istruzioni eseguite (in un secondo) del processore, indicato come MIPS o throughput, che dipende dal clock della pipeline e dal grado di scalarità del processore

# Pipeline

---

- Sia detto:
  - $T_s$  il tempo di esecuzione di una istruzione in un sistema senza pipeline
  - $T_p$  il tempo di esecuzione di una istruzione in un sistema con pipeline
- Si ha che:
  - $T_s \leq T_p$
  - Ciò avviene nel caso di pipeline imperfetto o nel caso di blocco del pipeline perfetto

# Valutazioni Prestazioni

---

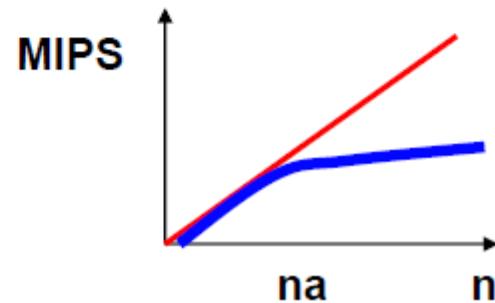
Prestazioni, valutate in MIPS:

**MIPS<sub>s</sub> = 1/T<sub>s</sub>** (sistema senza pipeline)

**MIPS<sub>p</sub> = n/T<sub>p</sub>** (sistema con pipeline), con n numero stadi

$$\text{MIPS}_s \ll \text{MIPS}_p$$

In teoria le prestazioni crescono linearmente con n, in realtà esiste un asindoto per  $n \geq n_a$  poiché al crescere di n non si riesce più a soddisfare il vincolo del pipeline ottimo.



# Valutazioni Prestazioni

---

Siano:

$k = \text{n. stadi pipeline}$

$\tau_i = \text{tempo dello } i\text{-esimo stadio}$

$\tau_{\max} = \max(\tau_i)$

Se si considerano  $N$  istruzioni, si ha:

1. **Tempo di esecuzione di una istruzione =  $k\tau_{\max}$**
2. **Tempo di esecuzione di  $N$  istruzioni, senza pipeline, =  $Nk\tau_{\max}$**
3. **Tempo di esecuzione di  $N$  istruzioni, con pipeline, =  $k\tau_{\max} + (N - 1) \tau_{\max}$**

( $k\tau_{\max}$  è tempo di esecuzione della prima istruzione che deve percorrere tutta pipeline)

# Valutazioni Prestazioni

---

**Fattore di accelerazione:**

(Tempo di esecuzione di N istruzioni, senza pipeline)/(Tempo di esecuzione di N istruzioni, con pipeline) =

$$Nk\tau_{max}/( k\tau_{max} + (N - 1)\tau_{max}) =$$

$$Nk/ (k + (N - 1))$$

*Si noti che per N grande il rapporto tende a k*

Ne deriva che i MIPS di picco di una architettura pipeline risultano:

$$\tau_{max} = m \text{ (periodi clock cpu)} = m / \text{(frequenza clock cpu)}$$

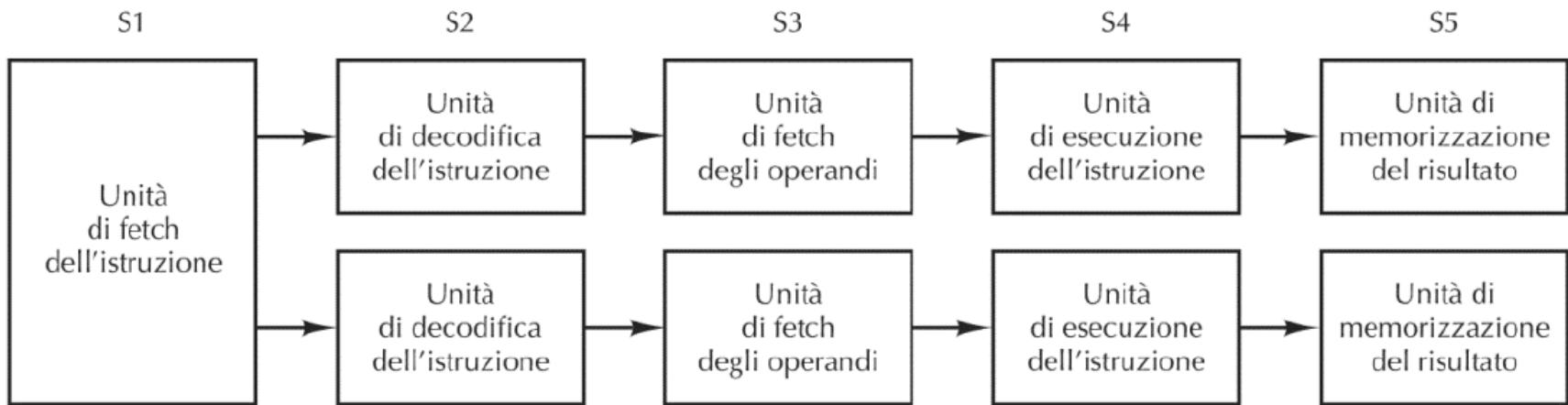
$$\text{MIPS} = 1/ \tau_{max} = (\text{frequenza clock cpu})/m$$

Essendo m il numero di periodi di clock richiesti per eseguire le operazioni in uno stadio

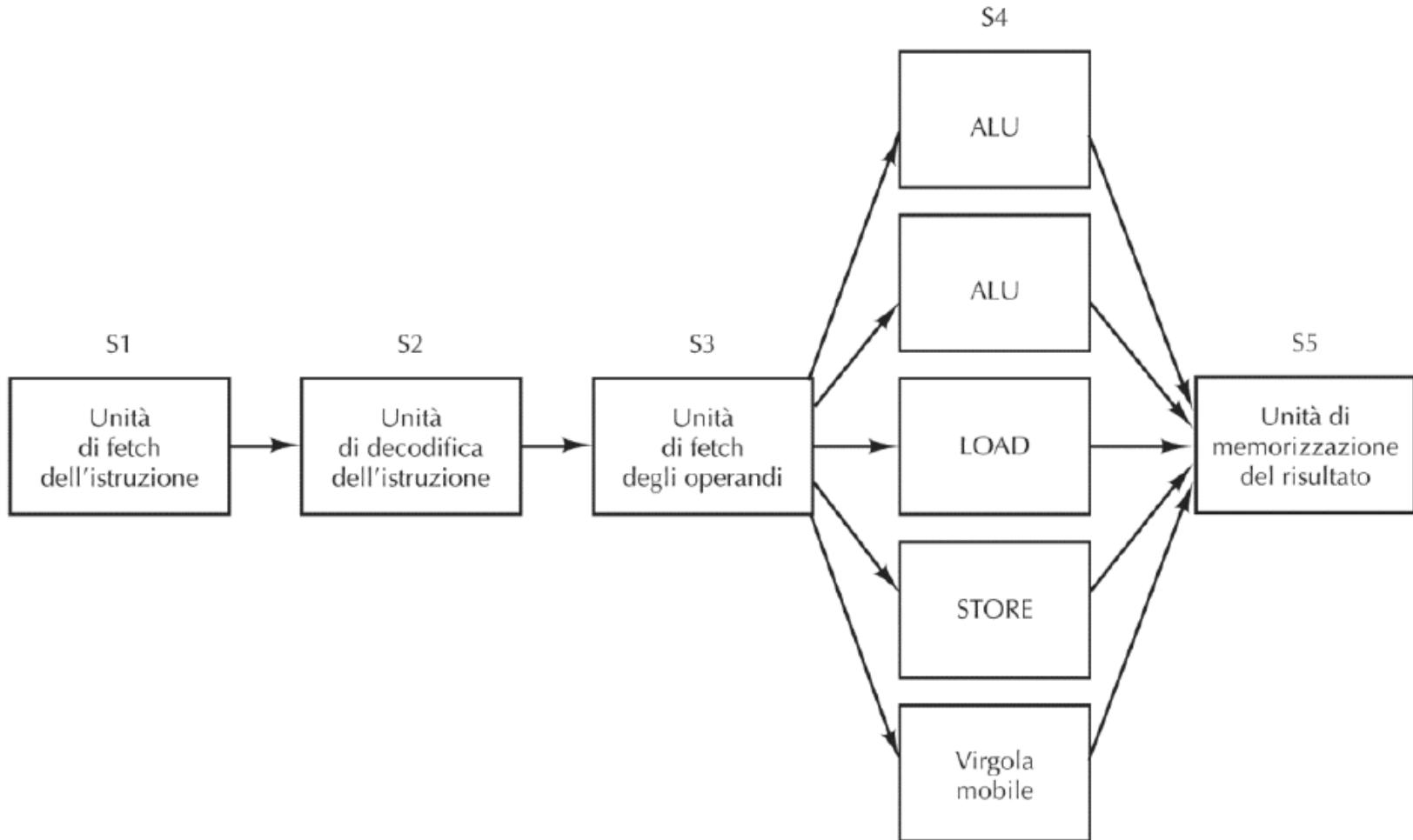
Nei processori RISC m in genere vale 1, ne deriva che un processore a 1 GHz ha MIPS = 1000 (istruzioni/s)

# Doppia Pipeline

---



# Architetture Superscalari



# Chip della CPU

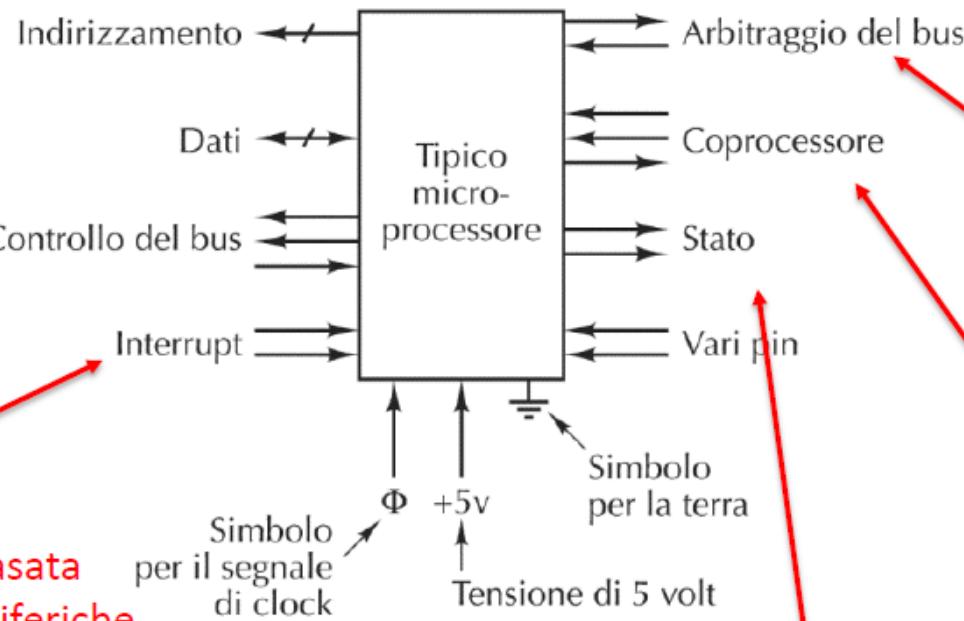
---

- Pin di tre tipi: Indirizzi, dati, segnali di controllo,
- Comune a tutte le CPU: alimentazione, segnale di clock
- Tipici segnali di controllo:
  - 1.Controllo del bus
  - 2.Arbitraggio del bus
  - 3.Interrupt
  - 4.Comunicazione con il coprocessore
  - 5.Stato
  - 6....

# Chip della CPU

Indicano quali operazioni sono richieste (es. Rd/Wr memoria)

Per gestione I/O basata su interrupt: le periferiche inviano un segnale quando hanno terminato una operazione di I/O richiesta dalla CPU



Altri controlli per reset, debugging, (lettura/scrittura stato), o per compatibilità con vecchi chip di I/O ...

Segnali necessari quando vi sono più componenti «attive» sul bus (es. controller DMA): occorre regolare l'accesso al bus

Nel caso vi siano coprocessori, (es. per l'aritmetica in floating point o per grafica) questi segnali permettono il coordinamento

# Un esempio di CPU: Intel Core i7

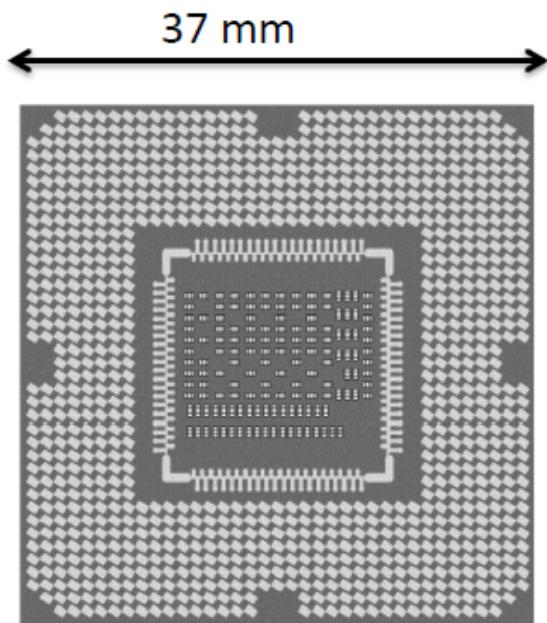


Figura 3.44 Disposizione fisica dei contatti del Core i7.

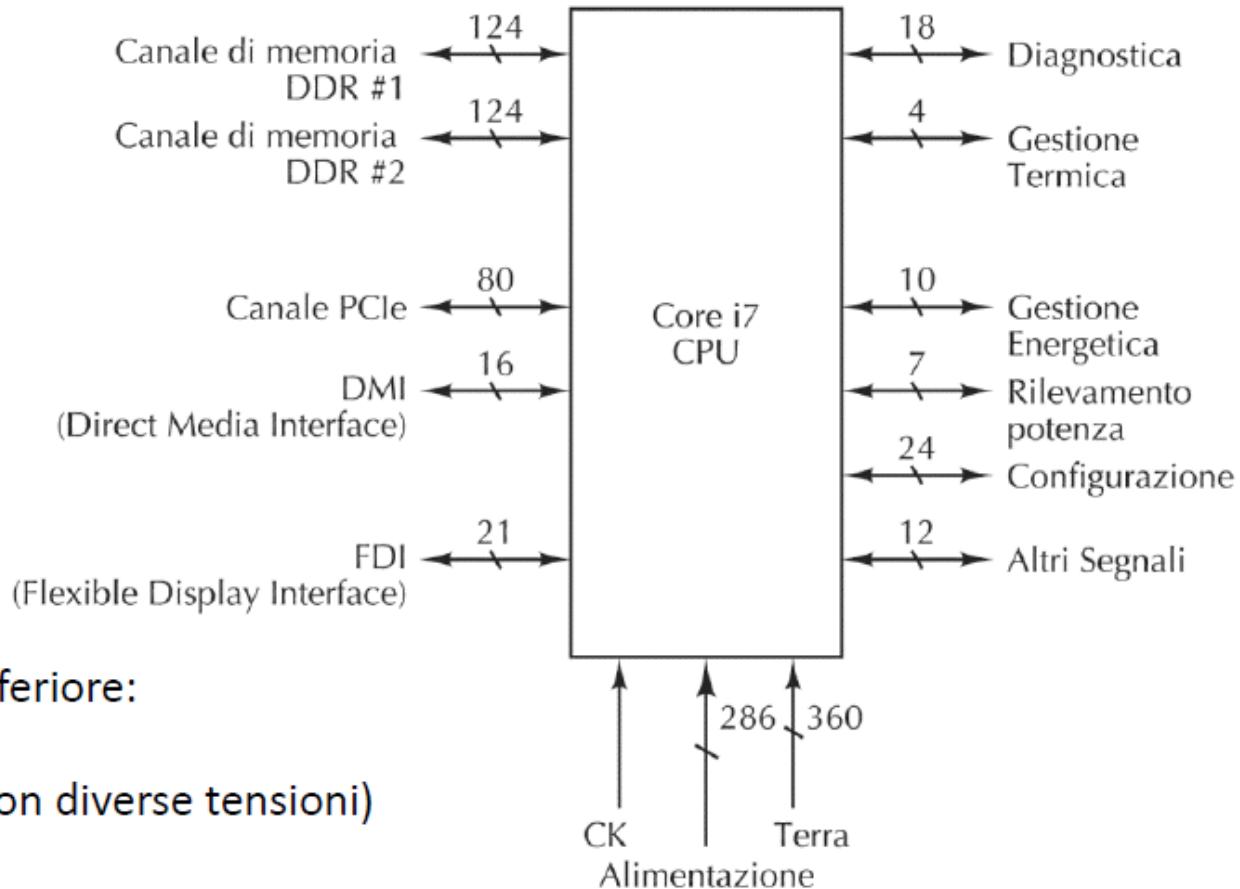


Figura 3.45 Disposizione logica dei contatti del Core i7.

# Bus

# Introduzione ai Bus

---

- Un bus è una struttura che interconnette due o più dispositivi.
- Un bus è una struttura condivisa: i valori che un dispositivo scrive sul bus sono accessibili a tutti gli altri dispositivi connessi.

# Struttura di un bus

---

- Un bus è composto da 3 gruppi di segnali:
  - segnali di dato: normalmente sono in numero pari ad un multiplo di 8; possono essere bidirezionali o unidirezionali (in tal caso è necessario un numero doppio di linee);
  - segnali di indirizzo: identificano lo slave con cui il master vuole comunicare (nonché quale parte dello slave è coinvolta);
  - segnali di controllo: forniscono informazioni di stato, di temporizzazione, di tipo (dei dati sul bus).

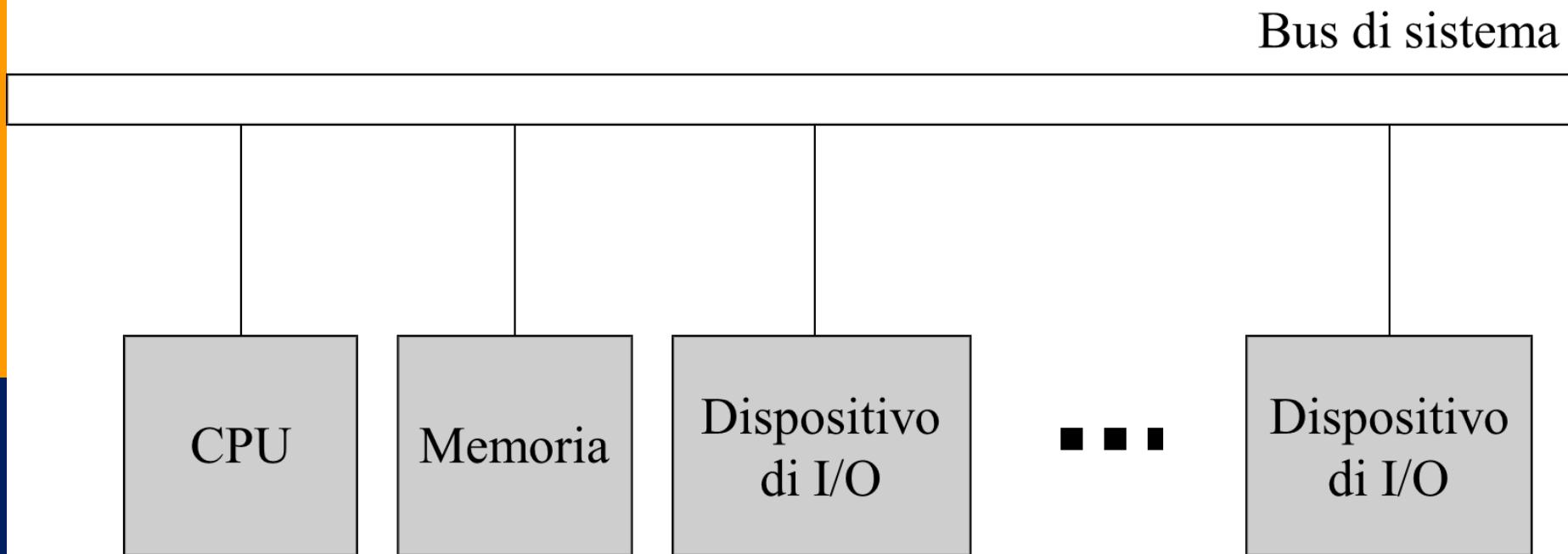
# Architetture di bus

---

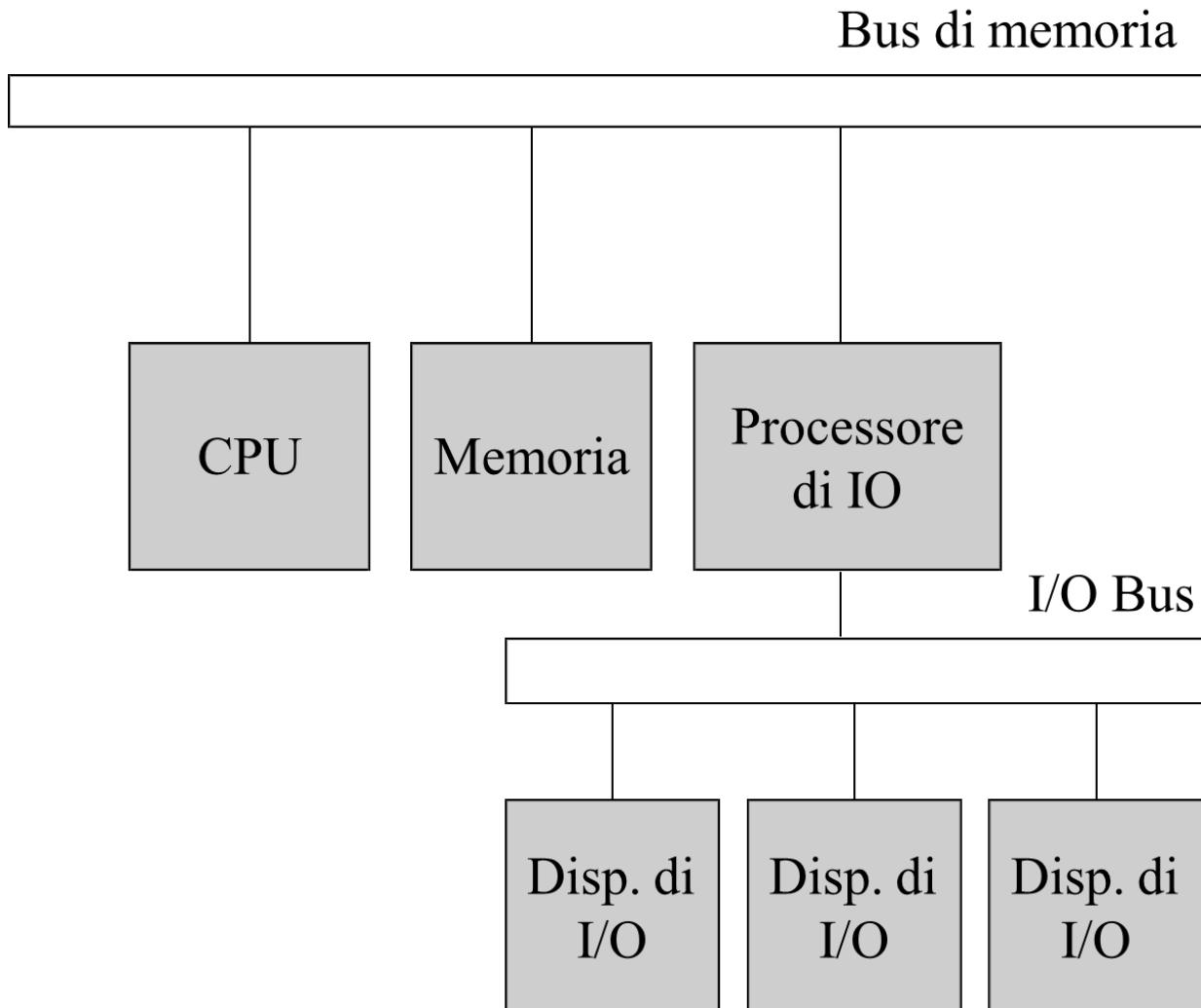
- Si possono avere 2 tipi di architetture a bus:
  - bus singolo: è la configurazione più semplice
  - bus multiplo: è utile laddove si desiderano prestazioni elevate, oppure quando si devono connettere diverse classi di dispositivi, con caratteristiche tra loro diverse.

# Bus singolo

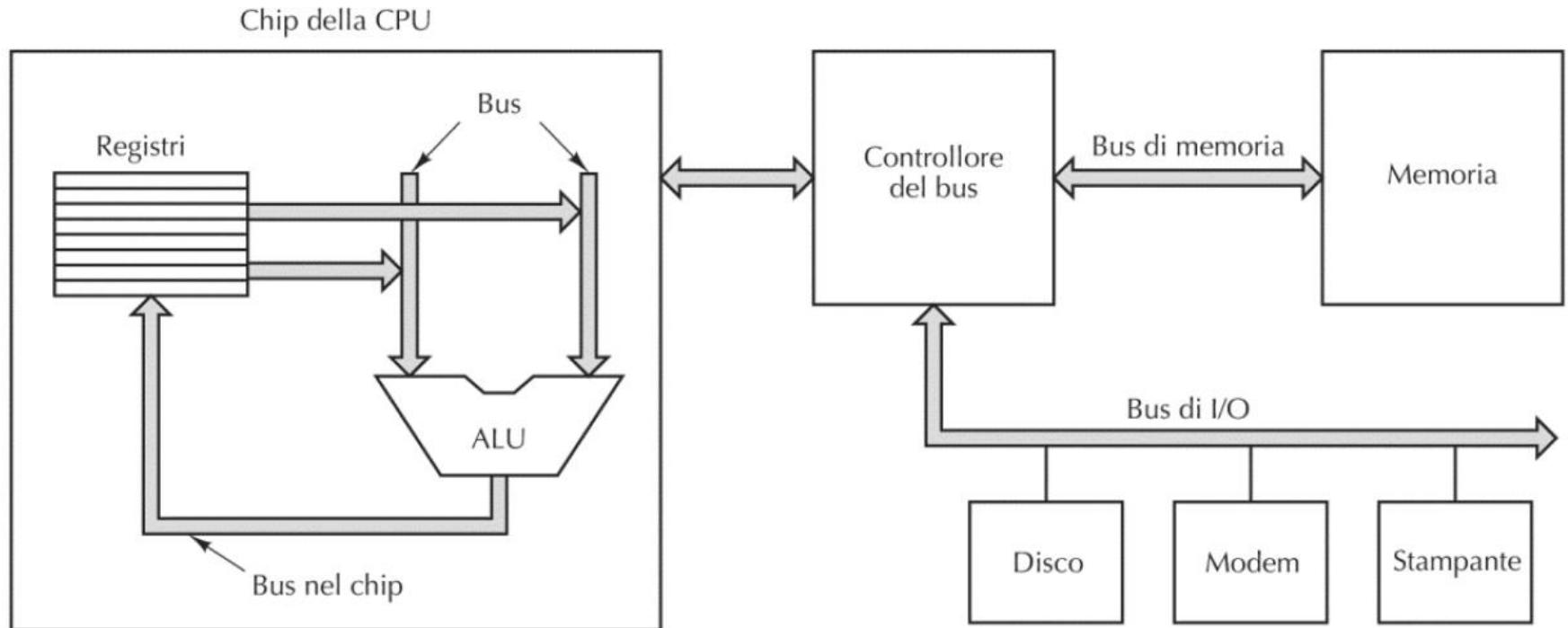
---



# Bus multiplo (esempio)



# Bus multiplo (esempio)



**Figura 3.35** Sistema di un calcolatore con più bus.

# Protocollo del bus

---

- insieme di regole di funzionamento che devono essere rispettate dai dispositivi ad esso collegati per poter comunicare.
- Componenti attive (master), cioè possono dare inizio ad un trasferimento dati, e passive (slave) che rimangono in attesa di una richiesta

# Master e slave

---

- In un sistema a bus le unità connesse sono di 2 tipi:
  - Unità master: inizia ogni procedura di trasferimento dati e sceglie lo slave con cui comunicare
    - Nei sistemi più semplici esiste un'unica unità master, che coincide con la CPU
    - nei sistemi più complessi esistono più unità master, e l'unità master cambia a seconda dei momenti
- unità slave: risponde ai comandi dell'unità master
  - le memorie e le interfacce dei periferici sono unità slave.

Master	Slave	Esempio
CPU	Memoria	Prelievo delle istruzioni e dei dati
CPU	Dispositivo di I/O	Inizio del trasferimento dei dati
CPU	Coprocessore	Passaggio dell'istruzione al coprocessore da parte della CPU
I/O	Memoria	DMA (Direct Memory Access)
Coprocessore	CPU	Prelievo degli operandi dalla CPU da parte del coprocessore

**Figura 3.36** Esempi di master e slave del bus.

# Principali caratteristiche dei bus

---

- Ampiezza del bus
- Temporizzazione
- Arbitraggio
- Operazioni consentite

# Aampiezza del bus

Numero di linee per indirizzi / dati: un numero di linee indirizzi ridotto può limitare la capacità di indirizzare memorie più grandi:

Bus ISA:

- XT 1981

larghezza bus

8 bit

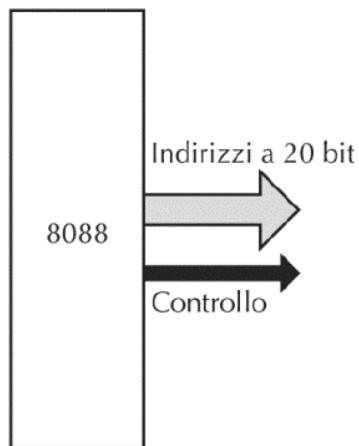
frequenza di  
clock 4.77  
MHz

- AT 1984

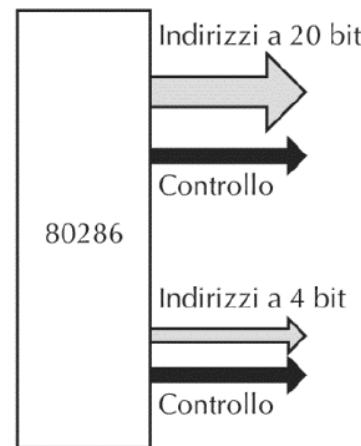
larghezza bus

16 bit

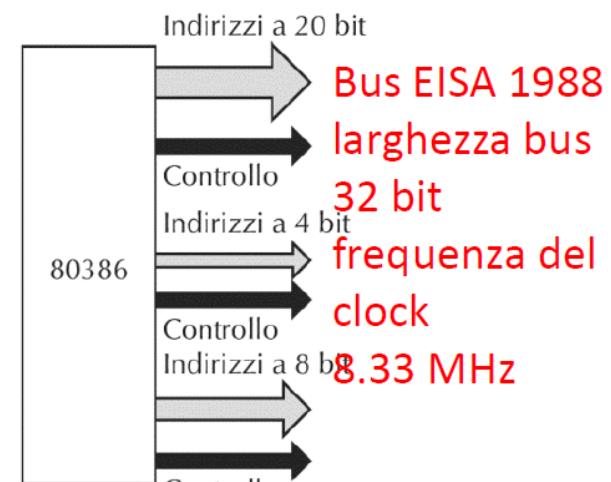
frequenza di  
clock 6 o 8  
MHz



(a)



(b)



(c)

Figura 3.37 Crescita nel tempo degli indirizzi del bus.

Banda max ISA: 16,7MB/sec  
max EISA: 33,3MB/sec

# EVOLUZIONE BUS

## EVOLUZIONE BUS

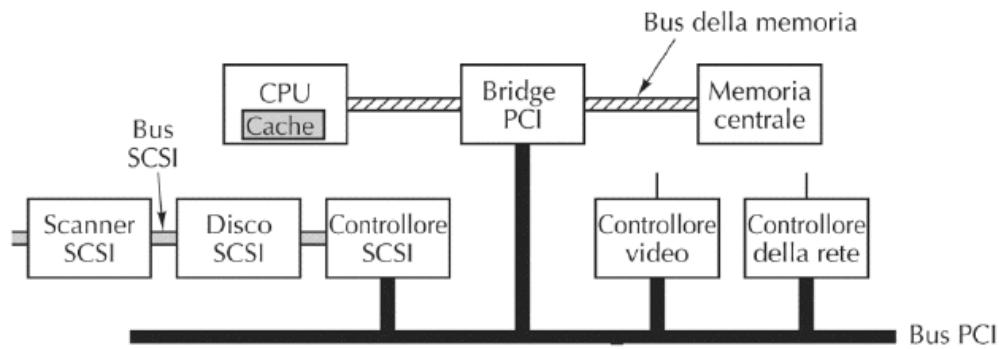


Figura 2.31 Tipico PC odierno con bus PCI. Il controllore SCSI è un dispositivo PCI.

**PCI Peripheral Component Interconnect  
66 MHz, trasf : 64 bit/ciclo = 528MB/sec**

**PCIe:** versione completamente rinnovata. Le connessioni sono seriali ma molto veloci (mentre PCI usava diverse ampiezze di bit: 8, 16, ...) e connessioni punto punto tramite commutatori. 1GB/sec per ogni «corsia» (si possono usare più corsie in parallelo)

Struttura del Calcolatore

**Un bus dedicato per mettere in comunicazione CPU e memoria rende più efficiente l'accesso in memoria.  
Se i controllori sono di tipo DMA possono accedere direttamente alla memoria.**

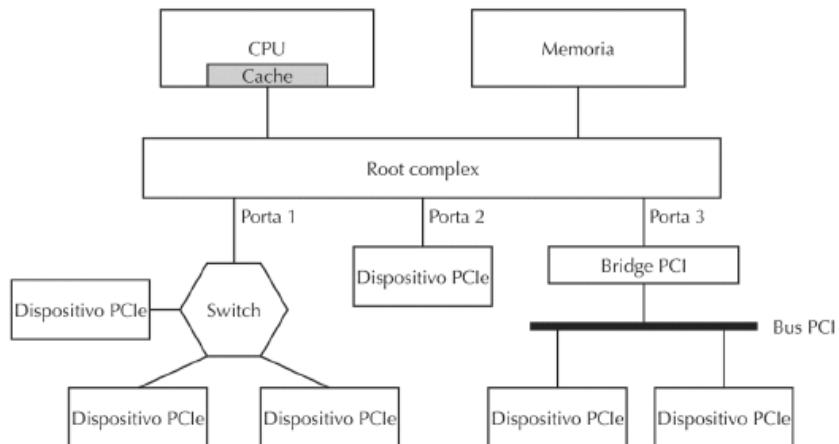


Figura 2.32 Esempio di architettura di un sistema PCIe con tre porte PCIe.

# AMPIEZZA DEL BUS

---

- ❑ Il numero di linee dati influenza sulla larghezza di banda.
  - ❑ A parità di linee dati, aumentando la frequenza del clock del bus si può aumentare la quantità di dati trasferiti nell'unità di tempo.
  - ❑ Un limite alla velocità del bus è posto dal problema del disallineamento del bus (linee distinte viaggiano a velocità leggermente differenti, fenomeno più marcato con l'incremento della velocità).
- ❑ Bus troppo ampi pongono anche problemi di spazio fisico: possibilità di utilizzare bus multiplexati

# Temporizzazione

---

L'utilizzo di un sistema a bus richiede la soluzione di 2 principali problemi:

- La definizione delle tempistiche con cui si svolgono le operazioni sul bus
- l'introduzione di un meccanismo per la gestione dei conflitti nell'accesso al bus.

# Temporizzazione

---

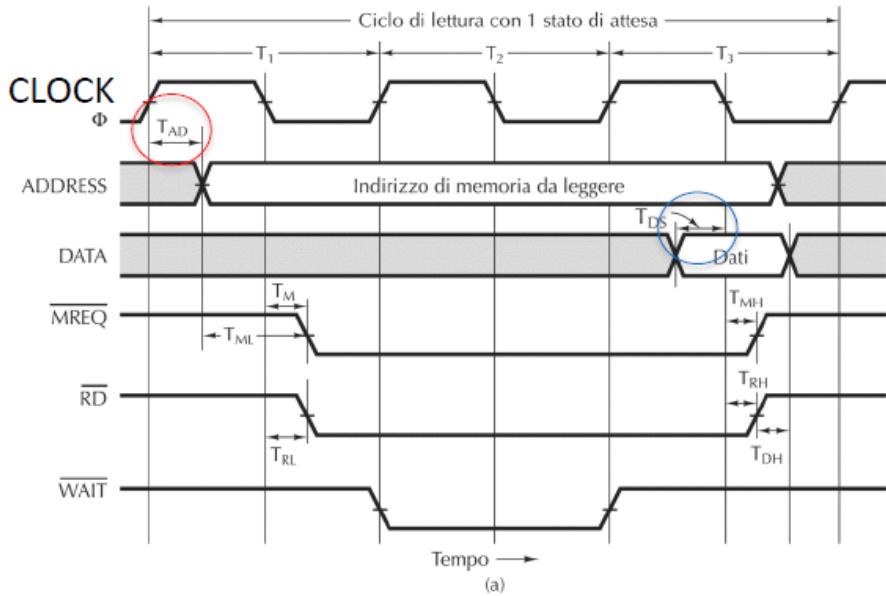
- ❑ Come possono le varie unità connesse ad un bus sapere quando leggere o scrivere dal/sul bus?
- ❑ Le soluzioni possibili sono riconducibili a due famiglie:
  - ❑ Bus sincroni
  - ❑ Bus asincroni.
- ❑ Le specifiche di un bus comprendono la descrizione del protocollo che i segnali dovranno seguire, nonché i limiti di tempo che dovranno essere rispettati.

# Bus sincrono

---

- Le unità sorgente e destinazione utilizzano lo stesso segnale di clock, che fa parte del bus stesso; alternativamente, le 2 unità possono avere clock separati, ma alla stessa frequenza, e scambiare periodicamente segnali di sincronizzazione
- la frequenza del clock è imposta dal dispositivo più lento
- ogni unità di dato è trasferita in un periodo di tempo prefissato (normalmente un periodo di clock)
- il meccanismo funziona bene su distanze ridotte.

# Bus sincrono



Simbolo	Parametro	Min	Max	Unità
$T_{AD}$	Ritardo dell'output dell'indirizzo		4	nsec
$T_{ML}$	Indirizzo stabile prima di MREQ	2		nsec
$T_M$	Ritardo di MREQ rispetto al fronte di discesa di $\Phi$ in $T_1$	3		nsec
$T_{RL}$	Ritardo di RD rispetto al fronte di discesa di $\Phi$ in $T_1$	3		nsec
$T_{DS}$	Tempo di impostaz. dei dati prima del fronte di discesa di $\Phi$	2		nsec
$T_{MH}$	Ritardo di MREQ rispetto al fronte di discesa di $\Phi$ in $T_3$	3		nsec
$T_{RH}$	Ritardo di RD rispetto al fronte di discesa di $\Phi$ in $T_3$	3		nsec
$T_{DH}$	Tempo di mantenimento dei dati dopo la negazione di RD	0		nsec

(b)

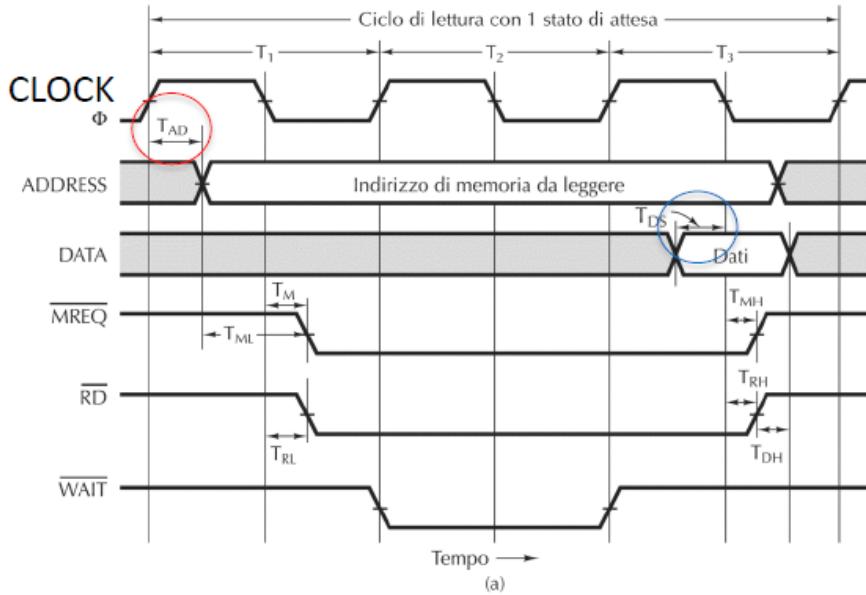
Es. di operazione di lettura: clock a 100MHz (ciclo da 10 nsec), inoltre a memoria impiega 15 nsec a fornire i dati (sono necessari 3 cicli = 30ns x lettura)

- $T_{AD}$  (tempo impiegato dalla CPU per inviare l'indirizzo dopo il fronte di salita di  $T_1 \leq 4$ nsec)
- $T_{DS} \geq 2$ nsec tempo in cui i dati rimangono stabili sul bus dati prima del fronte di discesa di  $T_3$
- $MREQ, RD$ , indicano alla memoria che si vuole accedere in lettura
- $WAIT$ , emesso dalla memoria all'inizio del ciclo  $T_2$ , indica alla CPU che il dato non sarà pronto entro quel ciclo; non è più asserito dopo la fine di  $T_2$  a indicare che entro  $T_3$  i dati arriveranno

*Nota: 25 – 4 – 2 = 19 ns tempo dalla comparsa dell'indirizzo all'invio dei dati*

Figura 3.38 (a) Temporizzazione di una lettura su un bus sincrono. (b) Specifiche di alcuni tempi

# Bus sincrono



Simbolo	Parametro	Min	Max	Unità
$T_{AD}$	Ritardo dell'output dell'indirizzo		4	nsec
$T_{ML}$	Indirizzo stabile prima di $MREQ$	2		nsec
$T_M$	Ritardo di $MREQ$ rispetto al fronte di discesa di $\Phi$ in $T_1$	3		nsec
$T_{RL}$	Ritardo di $RD$ rispetto al fronte di discesa di $\Phi$ in $T_1$	3		nsec
$T_{DS}$	Tempo di impostaz. dei dati prima del fronte di discesa di $\Phi$	2		nsec
$T_{MH}$	Ritardo di $MREQ$ rispetto al fronte di discesa di $\Phi$ in $T_3$	3		nsec
$T_{RH}$	Ritardo di $RD$ rispetto al fronte di discesa di $\Phi$ in $T_3$	3		nsec
$T_{DH}$	Tempo di mantenimento dei dati dopo la negazione di $RD$	0		nsec

(b)

- L'indirizzo dev'essere impostato almeno 2 ns prima che venga asserito  $MREQ$ ,
- I vincoli su  $T_M$  e  $T_{RL}$  garantiscono che  $MREQ, RD$ , vengano asseriti entro 3 ns dopo la discesa di  $T_1$ : in questo modo la memoria avrà tempo  $10+10-3-2=15$  nsec dopo che  $MREQ, RD$  sono asseriti.
- $T_{MH}$  e  $T_{RH}$  indicano il ritardo dalla generazione dei dati (fronte di discesa di  $T_3$ ) al momento in cui  $MREQ, RD$  non saranno più asseriti. Infine  $T_{DH}$  indica quanto tempo i dati devono rimanere stabili dopo che  $RD$  non è più asserito

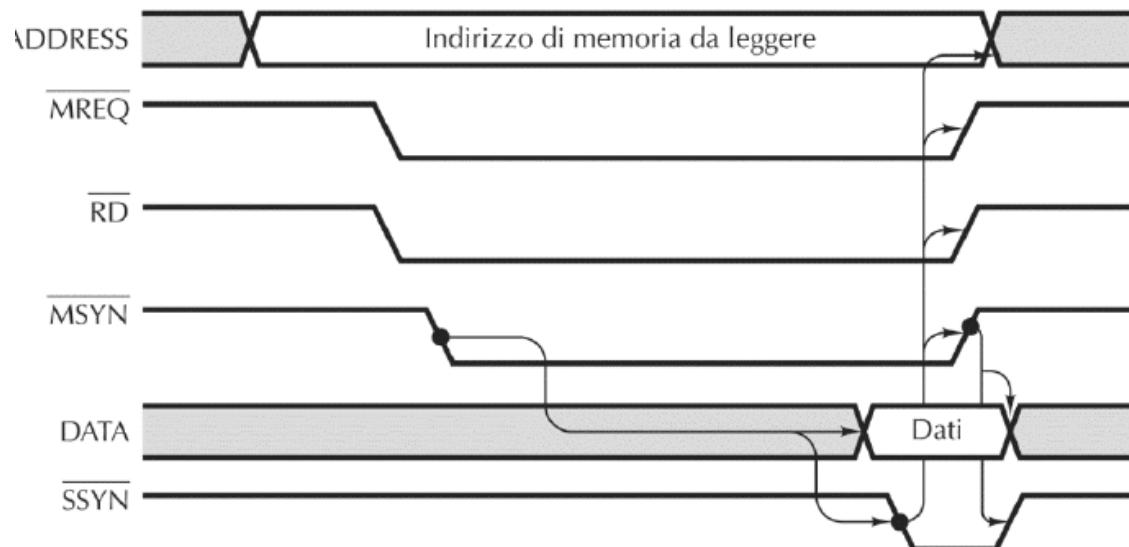
Figura 3.38 (a) Temporizzazione di una lettura su un bus sincrono. (b) Specifiche di alcuni tempi

# Bus asincroni

---

- Ogni operazione di comunicazione può avere una sua velocità, determinata da appositi segnali di controllo che accompagnano i segnali di dato ed implementano il cosiddetto hand shaking
- si ottiene così la massima flessibilità, a spese di una maggiore complessità del protocollo.

# Bus Asincrono



In questo caso non vi è alcun clock, e la sincronizzazione tra la CPU e la memoria avviene attraverso un **full handshake**, ovvero un coordinamento tra memoria e CPU, che inizia dopo che l'indirizzo è stabile sul bus come pure i segnali  **$\overline{MREQ}$**  e  **$\overline{RD}$** .

Quindi non ci sono precisi ritardi da rispettare, ma CPU e memoria si scambiano messaggi tramite una serie di segnali: prima dal master allo slave  $\overline{MSYN} \downarrow$ , una richiesta di lettura dati, e poi dallo slave al master  $\overline{SSYN} \downarrow$  per concludersi con un segnale di conferma di lettura dati  $\overline{MSYN} \uparrow$ , da parte del master seguito dal cambiamento di stato conclusivo dello slave esplicitato dal segnale  $\overline{SSYN} \uparrow$ , che libera il bus per una nuova operazione in memoria.

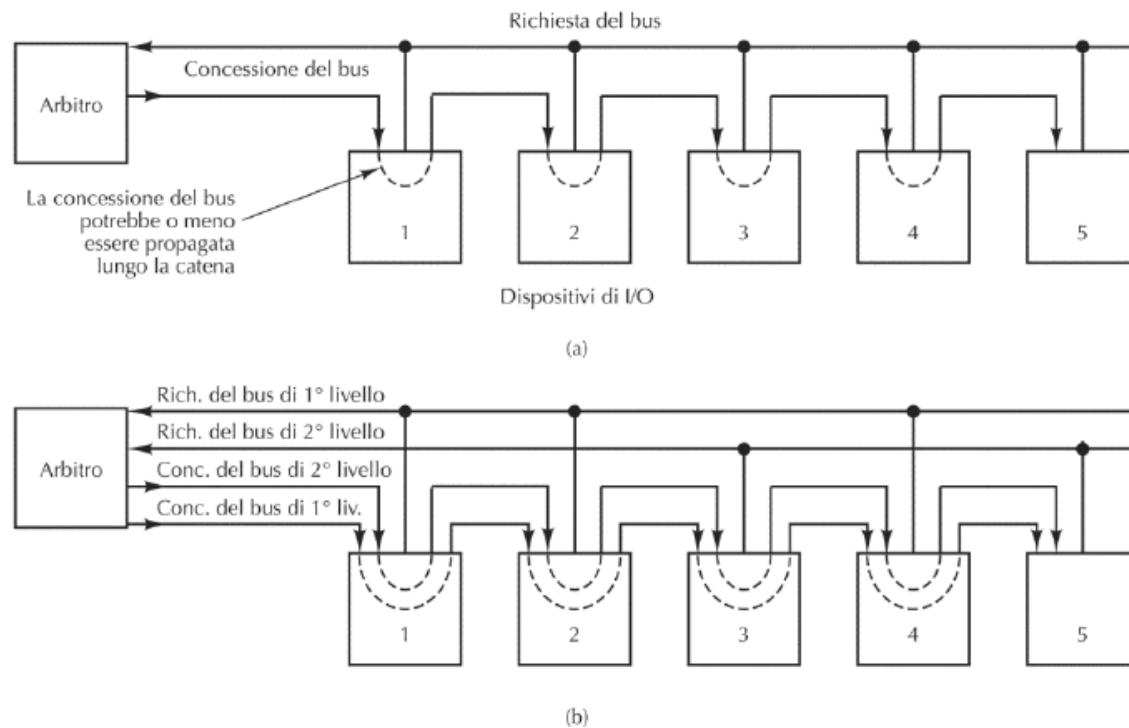
# Arbitraggio del Bus

---

- Necessario quando più master possono operare sullo stesso bus
- Può essere centralizzato o non centralizzato

# Arbitraggio centralizzato

## Arbitraggio centralizzato con collegamento di daisy chain a 1 e a 2 livelli di priorità

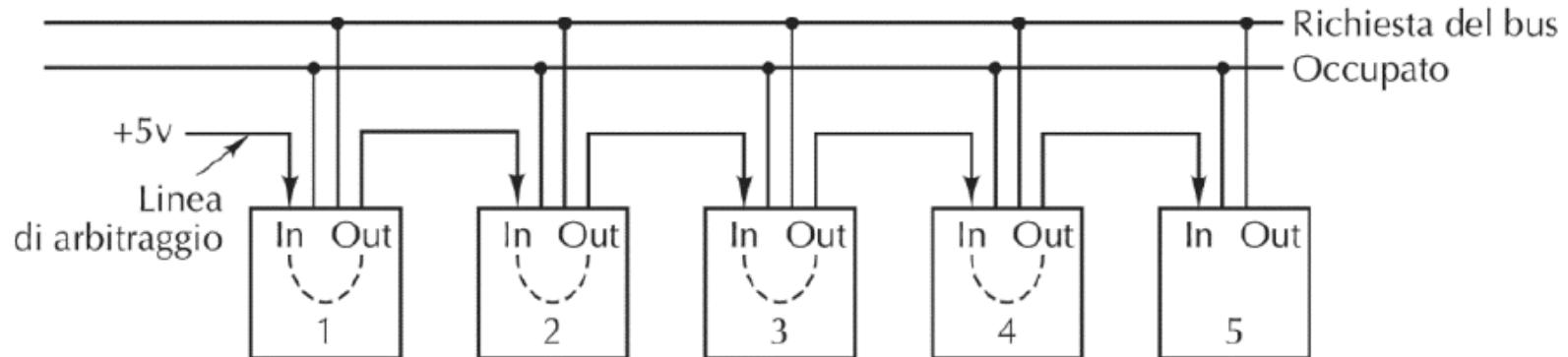


**Figura 3.40** (a) Arbitro del bus centralizzato e a un livello che utilizza un collegamento a festone.  
(b) Stesso arbitro, ma a due livelli.

# Arbitraggio non centralizzato

**Soluzione 1:** 1 linea separata per ogni potenziale master, tutti «sentono» le richieste di tutti gli altri, e quando più di un master asserisce la sua linea di richiesta di utilizzo del bus, se vede che nessuno con priorità più alta ha fatto la stessa richiesta acquisisce il bus e svolge l'operazione. Altrimenti attende il suo turno (Limite sul numero di dispositivi che si possono connettere, costo elevato).

**Soluzione 2:** simile al collegamento in daisy chain del metodo centralizzato ma senza arbitro. Le richieste sono in «or cablato». Quando chi ha fatto la richiesta riceve un segnale su «In», se vede che il bus non è Occupato, non asserisce Out e invece asserisce la linea «Occupato», altrimenti qualcuno a monte nella catena (priorità più alta) ha catturato il bus e il richiedente a priorità inferiore attende.



**Figura 3.41** Arbitraggio decentralizzato del bus.

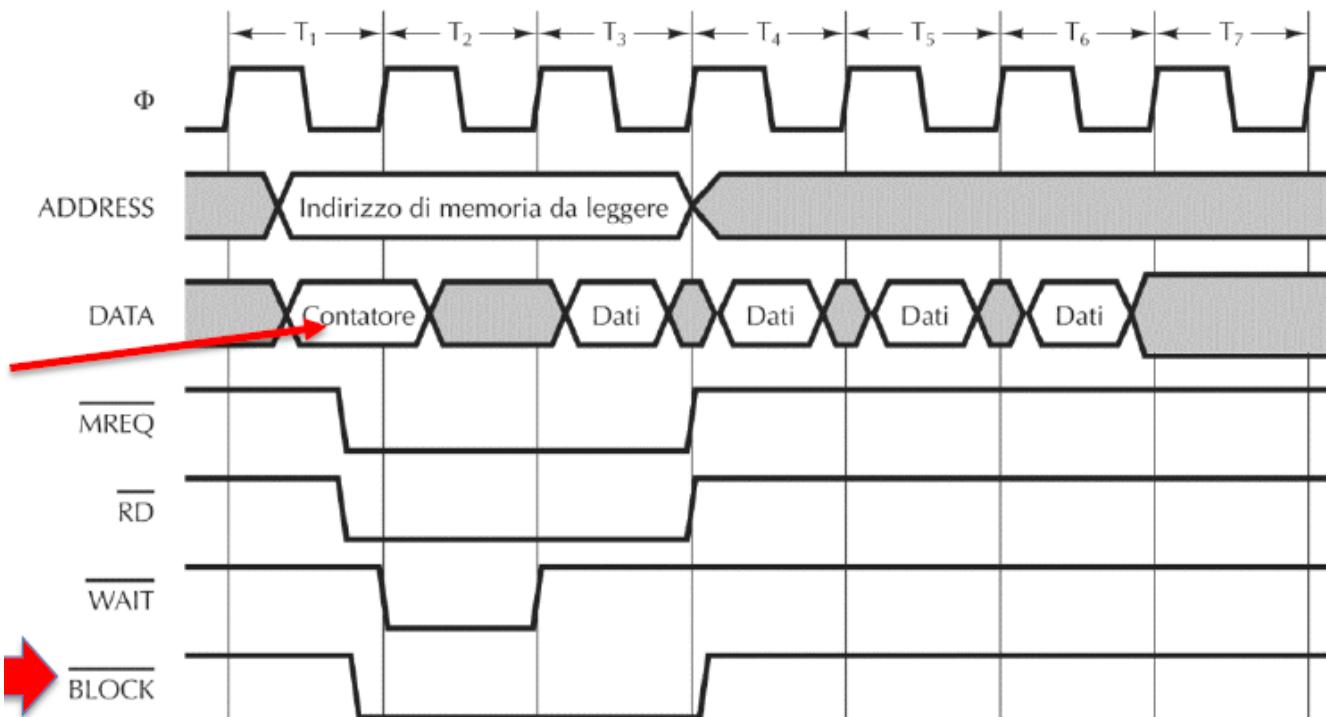
# Operazioni del Bus

---

- Le operazioni che si possono svolgere sul bus sono molteplici, oltre alla singola operazione di reade di writedi una parola, per esempio si può avere:
- Trasferimento di blocchi di dati che occupano più cicli
- Lock del bus per sincronizzazione di più CPU (accesso in mutua esclusione a dati condivisi in memoria)
- Gestione degli interrupt

# Trasferimento di blocchi dati

Per esempio quando si vogliono caricare in cache più parole consecutive: viene inviato l'indirizzo iniziale e un contatore, e viene asserito un segnale BLOCK per ottenere n parole in cicli consecutivi



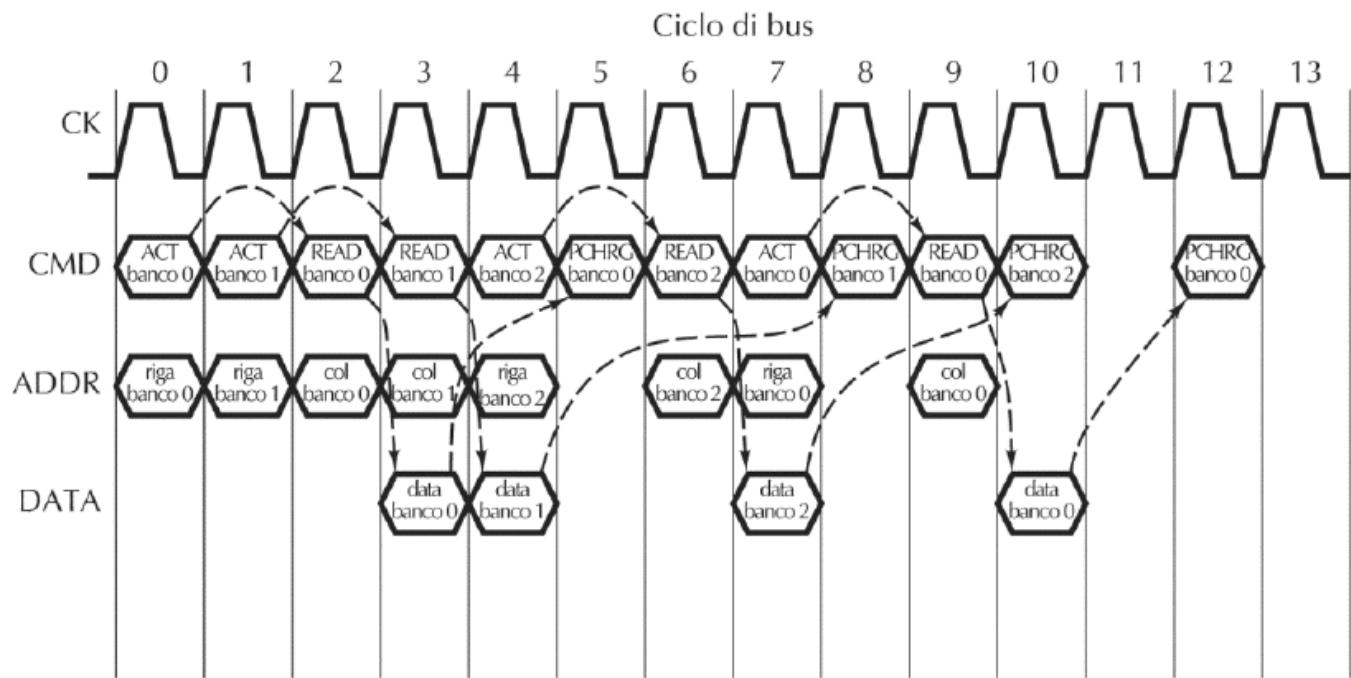
Nell'esempio si sono stati trasferiti 4 blocchi di dati in 6 cicli anziché in  $4 \times 3 = 12$  cicli.

# Pipeline di accessi in memoria

---

- In certe architetture è possibile applicare l'idea della pipeline agli accessi in memoria
- L'architettura Intel Core i7 può sovrapporre (parzialmente) gli accessi a banchi distinti di uno stesso chip di RAM
- L'interazione avviene in modo sincronizzato: la CPU conosce a priori i tempi di risposta della RAM ad ogni comando inviato.

# Pipeline di accessi alla memoria nel Core i7



I chip di memoria contengono più BANCHI che possono lavorare in parallelo: ciò consente di lavorare in pipeline (per sovrapposizione di operazioni su banchi distinti)

Figura 3.46 Richieste di memoria sull'interfaccia DDR3 del Core i7 gestite con pipeline.

Le richieste di memoria comprendono 3 fasi: ACTIVATE «apre» la riga di memoria per successivi accessi, RD/WR per accessi anche multipli a singole parole o a sequenze di parole della riga aperta, PRECHARGE «chiude» la riga precedentemente aperta per poter successivamente accedere ad un'altra. Tutto avviene in modo sincronizzato (tempi RAM sono noti alla CPU)

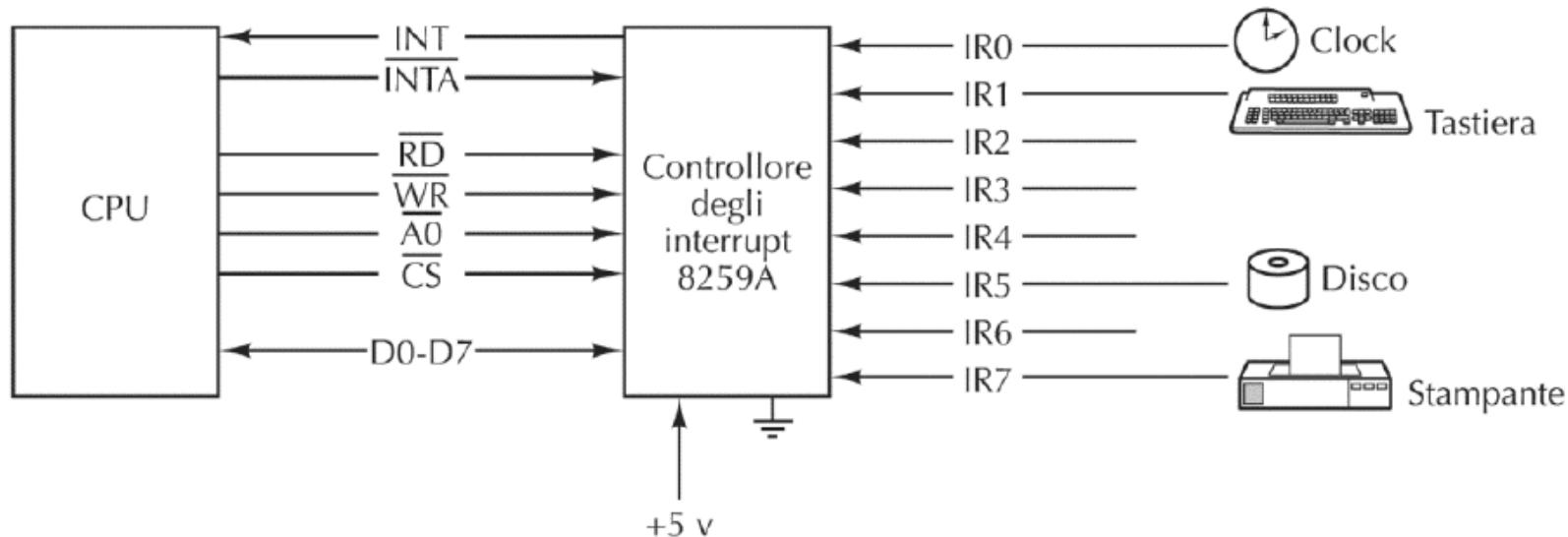
# Blocco del bus per accesso in mutua esclusione

---

- Ciclo speciale: Leggi –Modifica –Scrivi
- In un ambito multiprocessore garantisce che quando una CPU inizia il ciclo, questo non potrà essere interrotto, e la CPU potrà ottenere l'accesso in mutua esclusione ad una risorsa condivisa tra tutte le CPU, impostando a 1 la variabile di controllo (lock).
- Per evitare che due CPU possano interferire in questa fase di verifica del valore precedente del locke sua impostazione a 1, occorre che il bus rimanga bloccato per tutto il ciclo «Leggi lock», «Imposta locka 1», «Scrivi in memoria il valore di lock». Il valore restituito, è quello letto all'inizio del ciclo: la CPU ha possibilità di accedere se il valore restituito a 0, altrimenti deve ritentare tutta l'operazione.

# Gestione degli interrupt

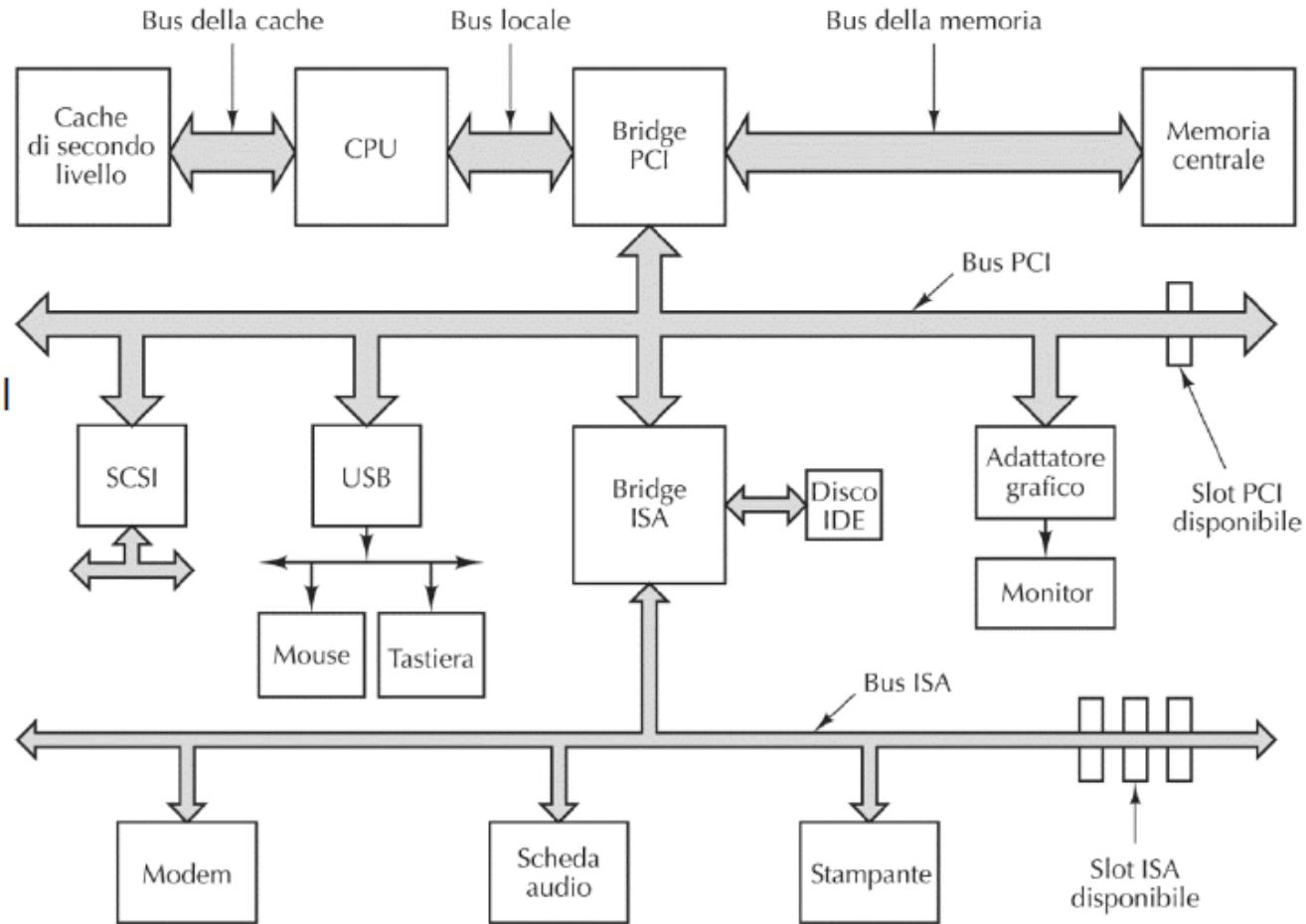
Quando la CPU richiede ad un dispositivo di I/O di eseguire una operazione, normalmente non rimane in attesa della fine della stessa, ma si dedica ad altri compiti. Quando l'operazione di I/O richiesta termina, il dispositivo invia un segnale di INTERRUPT alla CPU perché lo gestisca ed eventualmente invii una nuova richiesta al dispositivo. Il controllore degli interrupt controlla l'invio degli interrupt simultanei sulla base di un ordine di priorità. Il controllore segnala INT, la CPU risponde con  $\overline{\text{INTA}}$  non appena può gestirlo, il controllore invia l'ID del dispositivo a più alta priorità, che la CPU utilizza per richiamare l'*interrupt handler* appropriato.



# Struttura con bus multipli

Vi sono ancora periferiche compatibili con bus PCI anche se stato introdotto il più veloce PCIe (PCI Express)

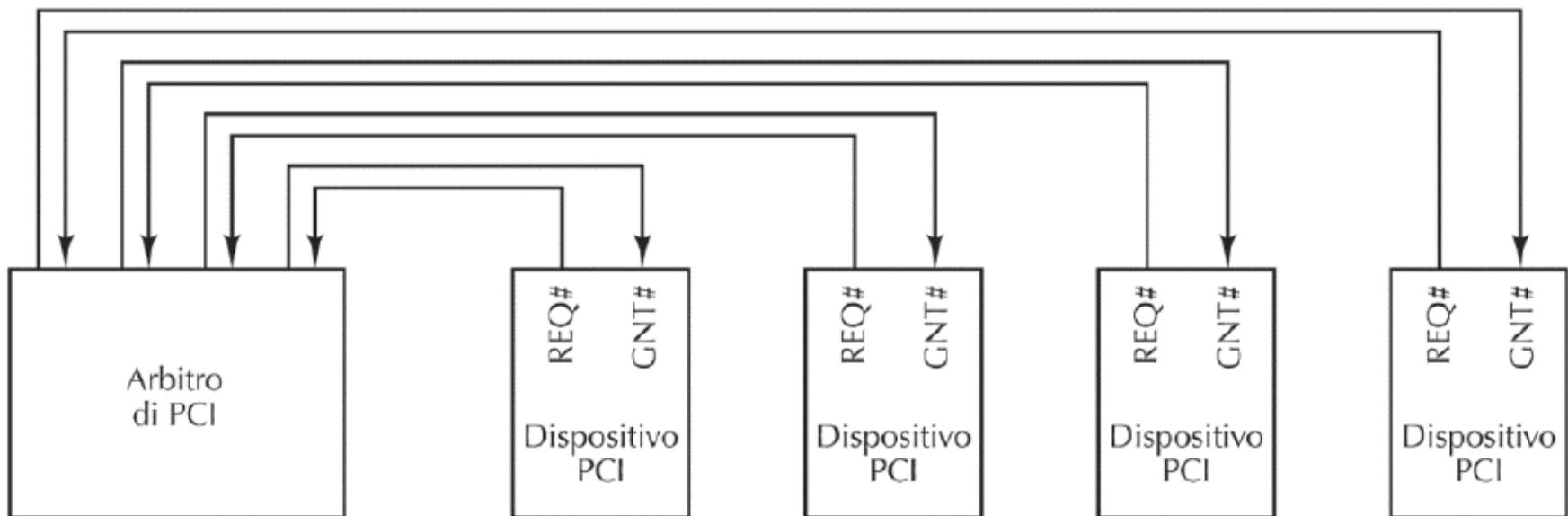
Il bus ISA oramai non è più utilizzato



**Figura 3.51** Architettura di uno dei primi Pentium. I bus più spessi hanno una maggiore larghezza di banda rispetto a quelli più sottili (la figura non è in scala).

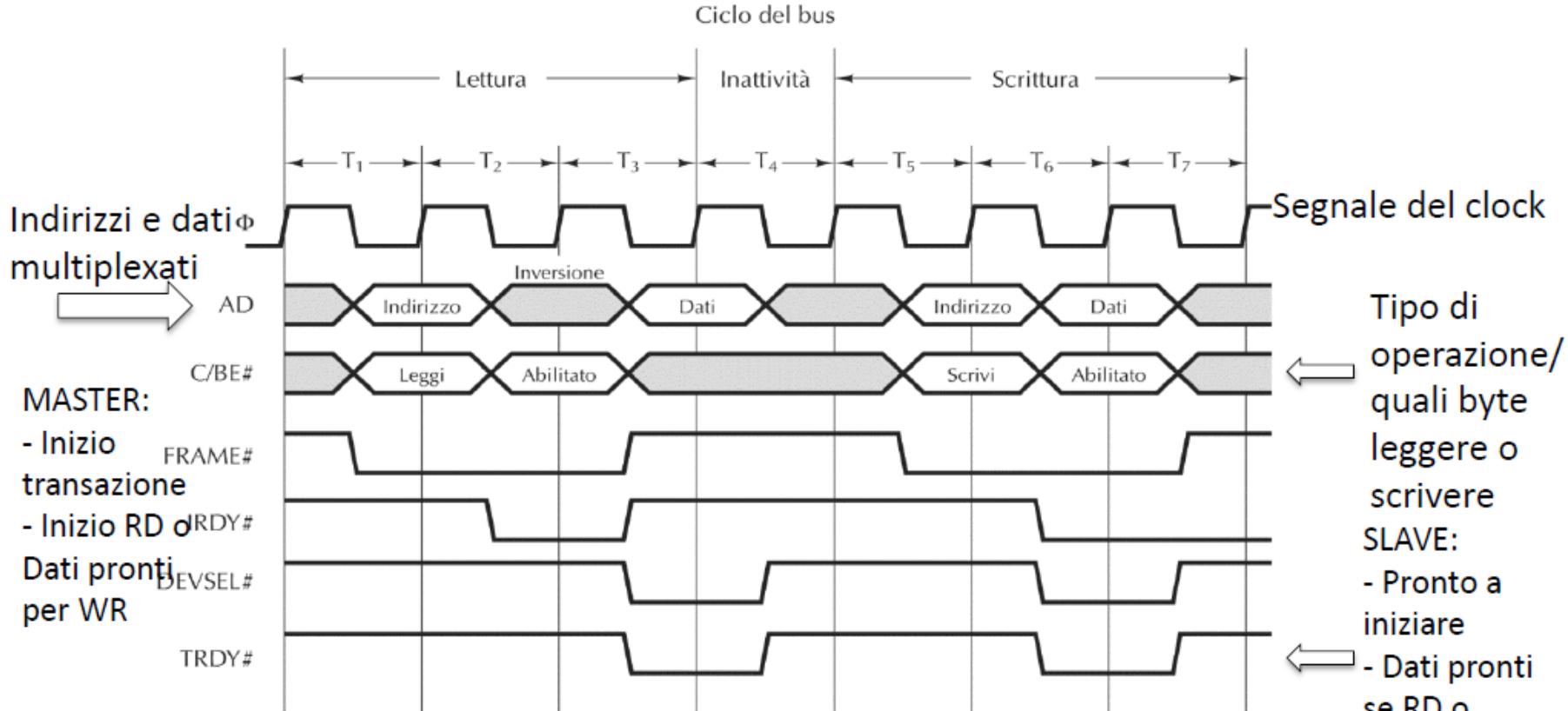
# Bus PCI: arbitraggio del bus

necessario perché più elementi possono accedere allo stesso bus



**Figura 3.53** Il bus PCI utilizza un arbitro del bus centralizzato.

# PCI: bus sincrono –esempio di funzionamento



**Figura 3.55** Esempi di transazioni di un bus PCI a 32 bit. I primi tre cicli sono utilizzati per un'operazione di lettura. Segue un ciclo d'inattività e infine tre cicli per un'operazione di scrittura.

Il simbolo # indica che il segnale asserito su valore basso ( $TRDY\#$  è equivalente a  $\overline{TRDY}$ )

# PCIe

- Si tratta di una rete d'interconnessione punto punto che supporta lo scambio di pacchetti tra i dispositivi. Molto più veloce di PCI (1GB/sec per corsia)
- Organizzato a livelli:

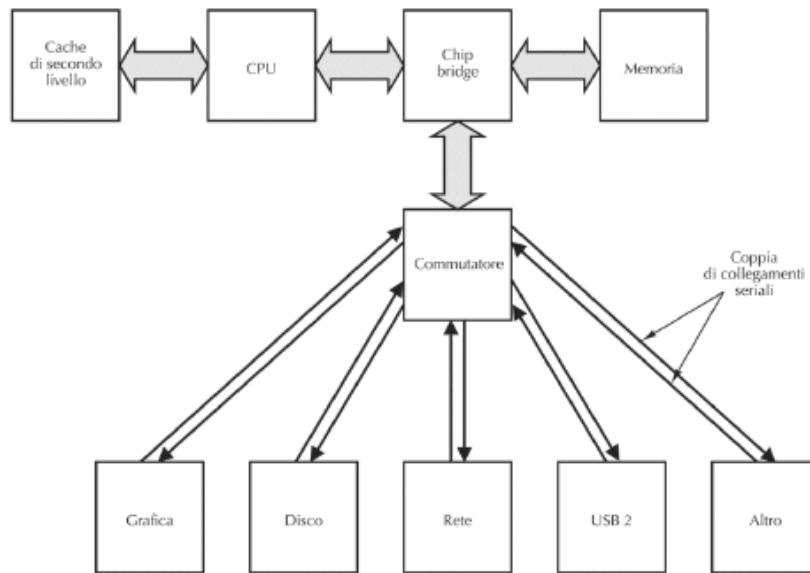


Figura 3.56 Tipico sistema PCIe.

Struttura del Calcolatore

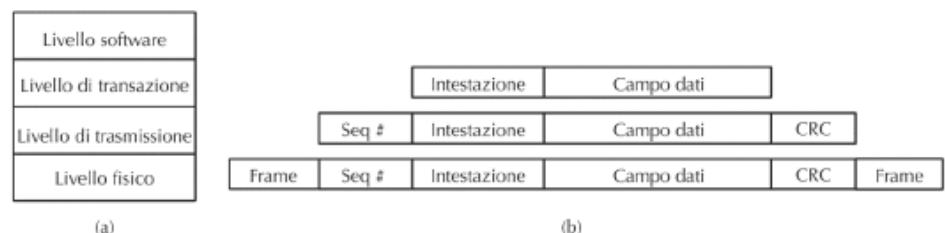
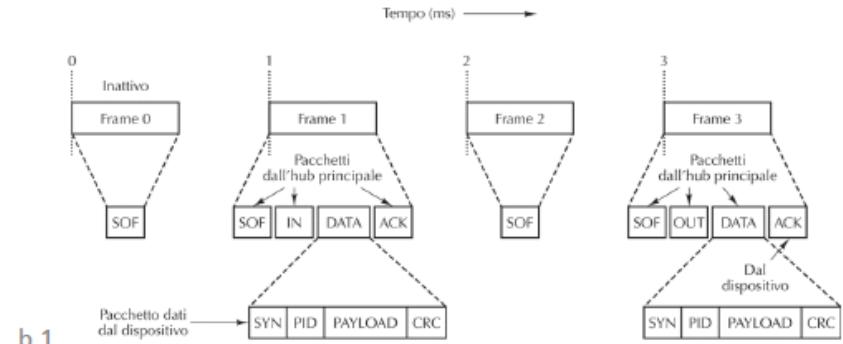
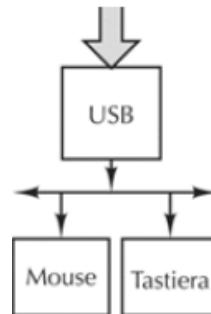


Figura 3.57 (a) Pila di protocolli di PCIe Express. (b) Formato di un pacchetto.

Sono implementati il controllo degli errori e il controllo del flusso (per adattare la comunicazione al più lento dei due interlocutori).

# USB (Universal Serial Bus)

- ❑ Orientato ai dispositivi lenti
- ❑ Uniformare interfaccia e permettere di poter collegare dispositivi senza aggiungere schede d'interfaccia e senza nemmeno riavviare il computer.
- ❑ Il PC (può) alimenta(re) il dispositivo attraverso il cavo di collegamento
  - ❑ USB1.0: 1.5 Mbps (tastiere, mouse)
  - ❑ USB1.1: 12Mbps (stampanti, fotocamere, HD esterni,
  - ❑ USB2.0: 480Mbs
  - ❑ USB3.0: 5Gbps



# Memorie (& rappresentazione Dati)

# Rappresentazione Interna Dei Dati

---

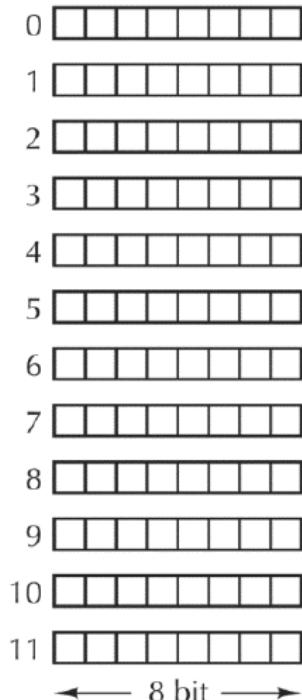
- **Tutte le informazioni** all'interno del computer sono rappresentate sotto forma di **sequenze di bit**

**BIT = Binary digIT** cifra binaria: valori possibili **0, 1** Per convenienza i bit possono essere raggruppati:

Nome	<b>Nibble</b>	<b>Byte</b>	<b>Word (parola)</b>
# bit	4 bit	8 bit=2 nibble	16/32/64 bit=2/4/8 byte
Esempio	0101	10110110	1100101100001111

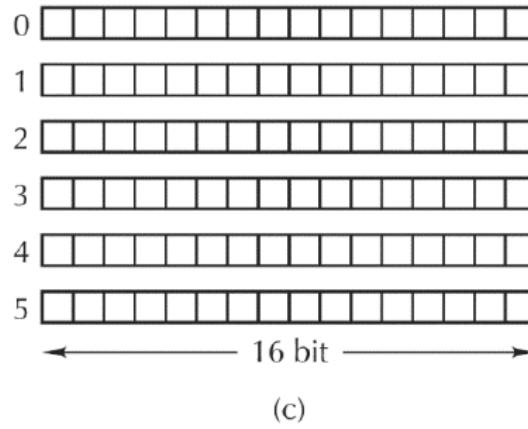
# Organizzazione della Memoria

Indirizzo



(a)

Indirizzo



Ci sono molti modi di raggruppare i bit di una memoria in celle di uguale dimensione: più grandi sono le celle, minore è il numero di indirizzi diversi necessari ad identificarle tutte (e meno bit sono necessari per esprimere gli indirizzi in binario).

***Normalmente le memorie attuali sono indirizzabili al byte.***

# Organizzazione della Memoria

Ipotizziamo di dover memorizzare un dato rappresentato su una parola di 4 byte in una memoria con celle di memoria di 1 byte: il dato occuperà 4 celle consecutive. Sono possibili due diversi ordinamenti dei byte che compongono la parola.

- LITTLE ENDIAN
- BIG ENDIAN

Esempio: **10001000**11101110**10101010****10111101**

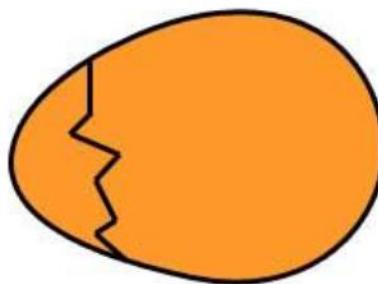
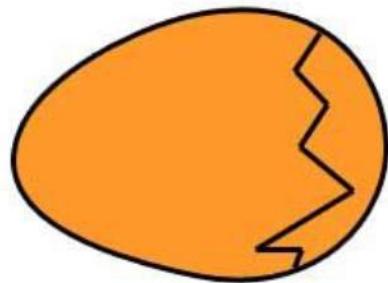
Inoltre si può allineare o meno la parola a indirizzi *i* multipli di 4

Indirizzo	Big Endian	Little Endian
i	<b>10001000</b>	<b>10111101</b>
i+1	11101110	<b>10101010</b>
i+2	<b>10101010</b>	11101110
i+3	<b>10111101</b>	<b>10001000</b>
i+4		
i+5		
.....		

# I viaggi di Gulliver

---

## I viaggi di Gulliver



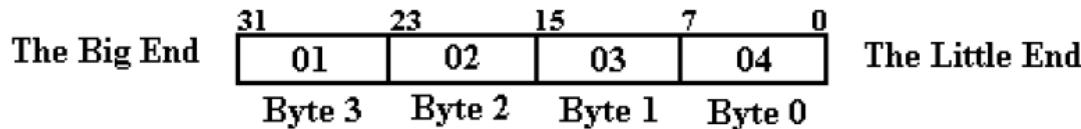
**BIG ENDIAN** - The way  
people always broke  
their eggs in the  
Lilliput land

**LITTLE ENDIAN** - The  
way the king then  
ordered the people to  
break their eggs

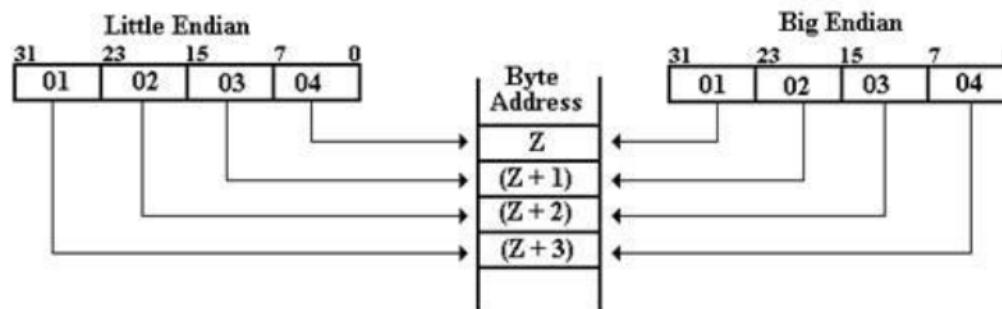
*Il nome dei due tipi di organizzazione dei dati multi-byte si riferisce ad un episodio contenuto ne «I viaggi di Gulliver» di Johnatan Swift dove si narra di grandi contrasti e rivolte nati all'interno del regno di Lilliput a causa di una legge che imponeva a tutti i sudditi di mangiare le uova rompendole dall'estremità più piccola anziché da quella più grande. Con riferimento ad un dato che occupa più byte il termine «little end» indica l'estremità corrispondente al byte meno significativo (più a destra), mentre il termine «big end» indica l'estremità opposta, corrispondente al byte più significativo.*

# Big endian vs. Little endian

diverse rappresentazioni in memoria del contenuto di un registro



Si vuole  
memorizzare il  
dato nella parola  
di memoria di  
indirizzo Z



Indirizzo	Z	Z+1	Z+2	Z+3
Rappresentazione LE	04	03	02	01
Rappresentazione BE	01	02	03	04

# Big endian vs. Little endian

---

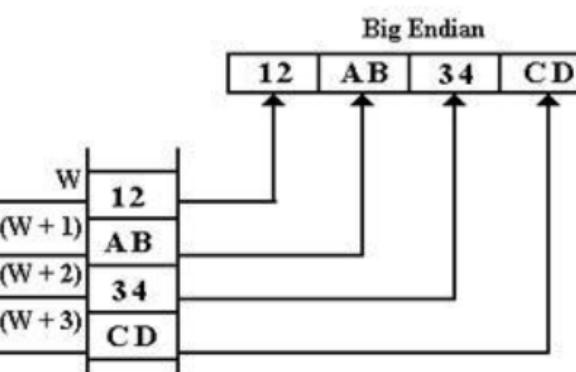
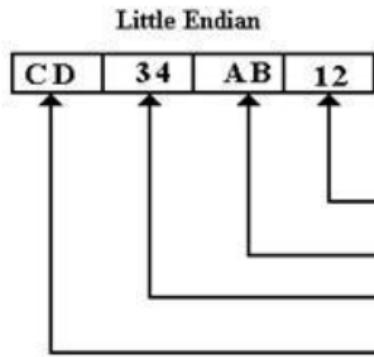
- I due diversi sistemi potrebbero portare disguidi nel caso in cui si vogliano trasferire dati tra due sistemi che utilizzano ordinamenti diversi.
- Consideriamo il seguente caso: il sistema A deve inviare al sistema B una sequenza di 40 byte che contiene la rappresentazione di 10 numeri interi, ciascuno codificato su una parola di 32 bit (= 4 byte).
- Se il sistema A usa un ordinamento Little Endian mentre B usa un ordinamento Big Endian cosa accade? B riconosce correttamente i «confini» dei 10 numeri, ma come interpreta il contenuto delle parole?

# Big endian vs. Little endian

Dato il seguente contenuto di alcuni byte in memoria, se carichiamo la parola di 4 byte (es. intero) di indirizzo W in un registro, alla fine cosa conterrà il registro? Dipende ... LE o BE ?

Indirizzo	$(W - 2)$	$(W - 1)$	W	$(W + 1)$	$(W + 2)$	$(W + 3)$	$(W + 4)$
Contenuto	0B	AD	12	AB	34	CD	EF

Interpretazione  
del sistema A



Interpretazione  
del sistema B

# Codice di protezione

---

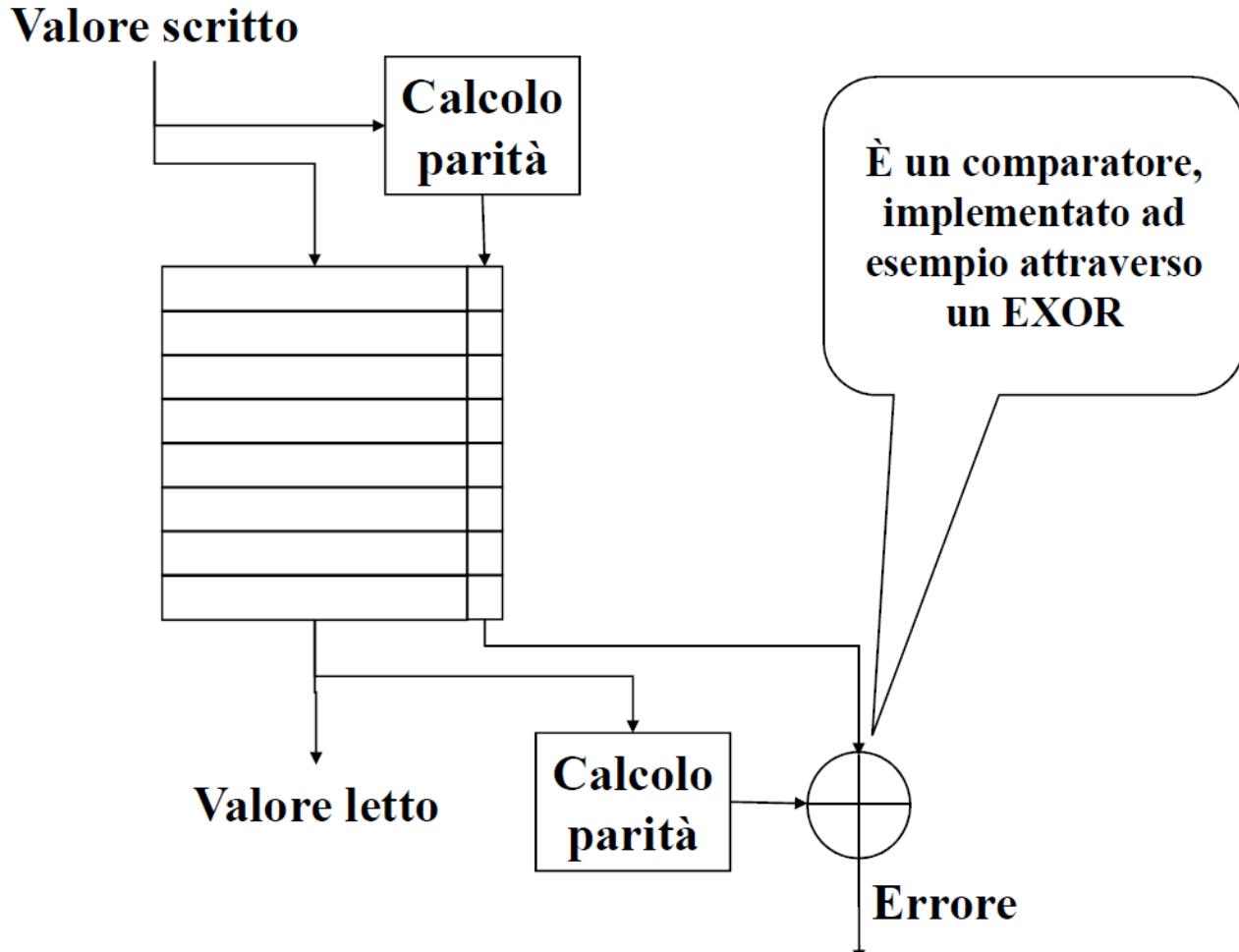
- Per aumentare l'affidabilità delle memorie, a ciascuna parola può essere associato un codice di protezione
- Un codice di protezione permette di utilizzare tecniche di rilevamento degli errori o tecniche di rilamento e correzione degli errori
- Vedremo un metodo di rilevazione degli errori, quello basato sul **bit di parità**, ed uno di correzione degli errori rilevati, il **codice Hamming**.

# bit di parità

---

- Il caso più semplice di codice di protezione è il codice di parità (1 bit)
- Funzionamento:
  - quando si scrive un valore nella parola, si calcola il relativo bit di parità, e lo si memorizza insieme al nuovo valore in un'apposita colonna
  - quando si legge la parola, si calcola il codice di parità associato al valore letto, e lo si confronta con quello memorizzato
  - in caso di diversità, si invia una segnalazione di errore.

# bit di parità



# bit di parità

---

- questo metodo permette di rilevare un errore, non di correggerlo;
- questo metodo introduce un bit aggiuntivo al dato vero e proprio; tutti i metodi di correzione e rilevazione degli errori aggiungono una serie di informazioni aggiuntive rispetto al dato di origine;

# codice Hamming

---

- Attraverso i codici di Hamming è possibile non solo rilevare, ma anche correggere eventuali errori verificatisi in una memoria.
- Verrà considerato solo il caso di codice di Hamming autocorrettivo per bit singolo (in grado cioè di correggere un eventuale errore su un solo bit).
  - permette di rilevare e correggere tutti gli errori singoli
  - permette di rilevare (ma non correggere) tutti gli errori doppi.
  - Per questo tale codice è detto SECDED (Single Error Correction Double Error Detection).

# Distanza di Hamming

---

- Una sequenza di  $n$  bit composta da  $m$  bit di dati e di  $r$  bit di controllo, con  $n=m+r$ , viene chiamata parola di codice (codeword) su  $n$  bit.
- Il numero di bit diversi tra due parole di codice viene detto **distanza di Hamming**; si può calcolare il numero di bit diversi facendo l'*or esclusivo* delle due stringhe e contando il numero di bit 1 del risultato; se due parole hanno distanza  $d$  significa che servono  $d$  errori per trasformare una nell'altra.
-

# Distanza di Hamming

---

- In genere non tutte le possibili stringhe di  $n$  bit ( $2^n$ ) sono legali (anche se quelle con  $m$  bit di dati lo sono);
  - la distanza minima tra le parole legali del codice è la distanza di Hamming del codice.
- La distanza di Hamming indica quanti errori si possono rilevare e quanti se ne possono correggere.
- Per rilevare  $d$  errori serve una distanza di  $d+1$ ; per correggere  $d$  errori serve una distanza di  $2d+1$  (la parola originale è la più vicina valida).



# Codifica dei caratteri

---

- Per poter codificare in binario informazioni testuali occorre definire un codice di rappresentazione dei caratteri
- Il primo codice di rappresentazione dei caratteri standardizzato negli anni `60 ma in uso ancora oggi (con alcune varianti) è il codice ASCII (American Standard Code for Information Interchange)
- Il codice ASCII base è di 7 bit (a cui in origine veniva aggiunto un bit di parità in fase di trasmissione), ma ne esiste uno esteso (anzi più d'uno) su 8 bit.

# ASCII base

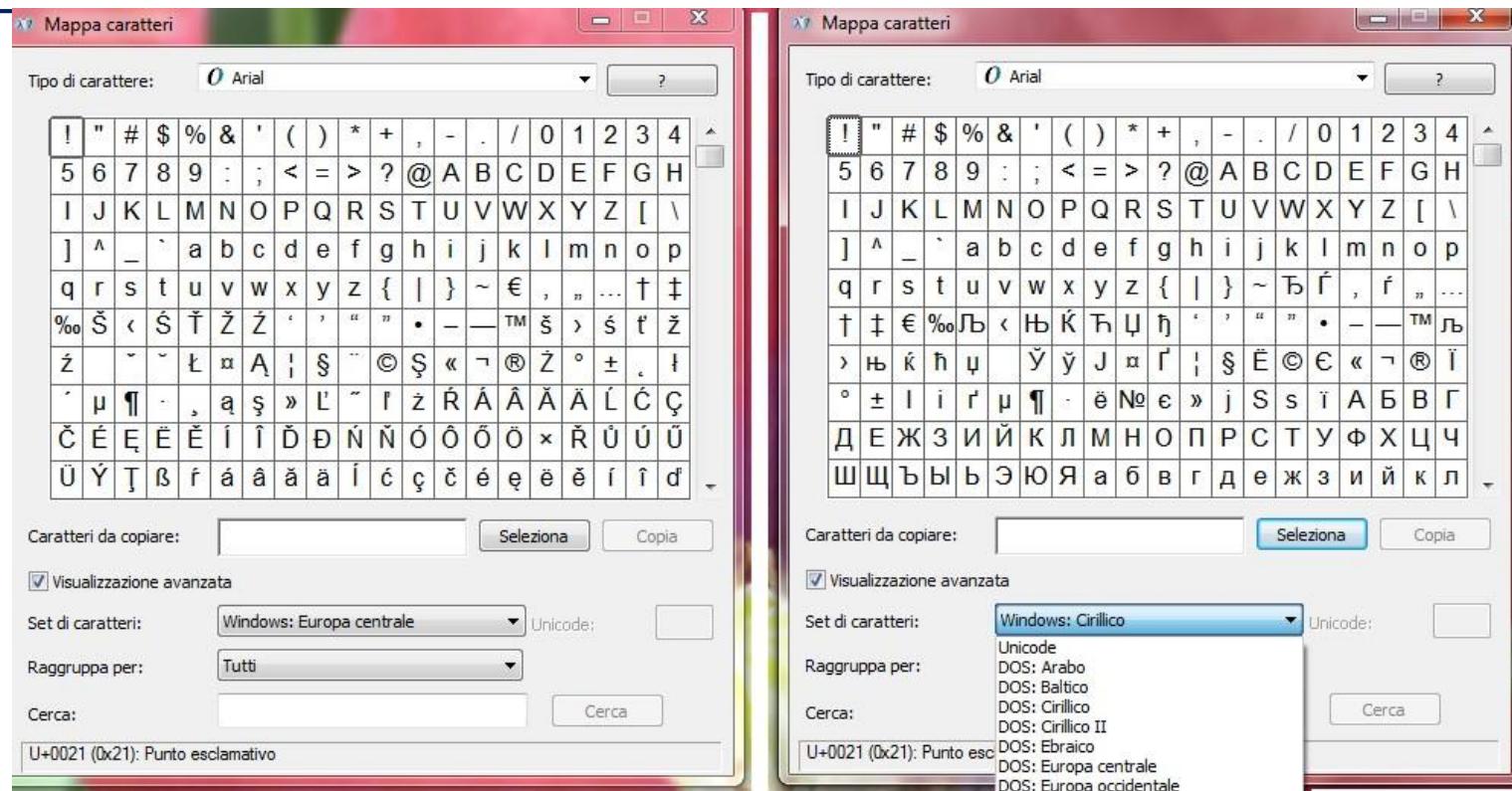
- I primi 32 codici sono riservati a caratteri di controllo utilizzati nella comunicazione tra il terminale e il sistema centrale.

I successivi 96 codici rappresentano le lettere dell'alfabeto inglese (26) minuscole e maiuscole, le cifre numeriche da 0 a 9, punteggiatura, parentesi, e altri caratteri di uso corrente.

Esa	Nome	Significato	Esa	Nome	Significato
0	NUL	Nullo	10	DLE	Uscita trasmissione (Data Link Escape)
1	SOH	Inizio intestazione (Start Of Heading)	11	DC1	Controllo periferica 1
2	STX	Inizio testo (Start Of Text)	12	DC2	Controllo periferica 2
3	ETX	Fine testo (End Of Text)	13	DC3	Controllo periferica 3
4	EOT	Fine trasmissione (End Of Transmission)	14	DC4	Controllo periferica 4
5	ENQ	Interrogazione (Enquiry)	15	NAK	Riconoscimento negativo (Negative AcKnowledgement)
6	ACK	Riconoscimento (ACKnowledgement)	16	SYN	Annulla (SYNchronous Idle)
7	BEL	Campanello (BELL)	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Tabulazione orizzontale (Horizontal Tab))	19	EM	Fine supporto (End of Medium)
A	LF	Riga nuova (Line Feed)	1A	SUB	Sostituisci (SUBstitute)
B	VT	Tabulazione verticale (Vertical Tab)	1B	ESC	Esc (ESCAPE)
C	FF	Avanzamento carta/nuova pagina (Form Feed)	1C	FS	Separatore di file (File Separator)
D	CR	Ritorno a capo (Carriage Return)	1D	GS	Separatore di gruppi (Group Separator)
E	SO	Disinserzione (Shift Out)	1E	RS	Separatore di record (Record Separator)
F	SI	Inserzione (Shift In)	1F	US	Separatore di unità (Unit Separator)

Esa	Car	Esa	Car	Esa	Car	Esa	Car	Esa	Car
20	Spazio	30	0	40	@	50	P	60	'
21	!	31	1	41	A	51	Q	61	a
22	"	32	2	42	B	52	R	62	b
23	#	33	3	43	C	53	S	63	c
24	\$	34	4	44	D	54	T	64	d
25	%	35	5	45	E	55	U	65	e
26	&	36	6	46	F	56	V	66	f
27	'	37	7	47	G	57	W	67	g
28	(	38	8	48	H	58	X	68	h
29	)	39	9	49	I	59	Y	69	i
2A	*	3A	:	4A	J	5A	Z	6A	j
2B	+	3B	;	4B	K	5B	[	6B	k
2C	,	3C	<	4C	L	5C	\	6C	l
2D	-	3D	=	4D	M	5D	]	6D	m
2E	.	3E	>	4E	N	5E	^	6E	n
2F	/	3F	?	4F	O	5F	_	6F	o
								7F	DEL

# ASCII esteso



Esistono diverse estensione dell'ASCII base: ogni estensione è chiamata code page. Quando si codifica un testo occorre specificare quale code page si sta utilizzando, e non si possono inserire caratteri da code page diverse nello stesso testo.

# UNICODE

---

Lo standard Unicode è stato introdotto per standardizzare la codifica di caratteri di testo in sistemi informatici, si associa allo standard internazionale UCS (Universal Character Set) ISO/IEC 10646 che definisce possibili codifiche concrete per i codepoint.

Lo standard è reperibile al sito:

<http://www.unicode.org/>

Unicode propone uno standard per assegnare codici univoci (chiamati codepoint) a caratteri e simboli di tutti i linguaggi scritti, simboli matematici etc . . .

Unicode (prima versione) usa un codice a 16 bit, con cui è possibile codificare

$2^{16} = 65536$  caratteri distinti (in esadecimale 0 ... FFFF)

In verità anche questo numero così grande non è stato sufficiente a soddisfare tutti, e nel 1996 sono stati introdotti 16 nuovi *plane* ciascuno basato su codifica a 16 bit (per un totale di 1 114 112 codepoint). Il piano 0 si chiama BPM: Basic Multilingual Plane

# UNICODE

---

- I codepoint di UNICODE definiscono codici univoci per moltissimi caratteri: oltre all'estensione già menzionata ne sono state introdotte altre (prevedendo la possibilità di utilizzare 32 bit per ciascun codepoint – la versione attuale è la 11.0)
  
- Sono rappresentate tantissime lingue esistenti, ma anche lingue morte, simboli vari (come gli emoji), sono state proposte anche lingue fantastiche (elfico)

# UNICODE

home.unicode.org

App Old repository journal utile\_lavoro Studio Vedere Architettura Cloud parigi attuale Progetti | Altri Pref.

UNICODE

U+3001 U+067B U+0B37 U+266B U+1557 U+FF64 U+1F3EF U+0A68 U+00F0 U+0FSD

Y U+01D0 U+055E U+0F0D U+FE3F U+10EC U+2704 U+05E9 U+069C U+FF9E U+0939

Adopt a Character +

Emoji +

Basic Info +

News

Events

Connect +

Membership +

Press

U+13EA U+2663

U+26AC U+FF10

Everyone in the world should be able to use their own language on phones and computers.

LEARN MORE ABOUT UNICODE

OFFICIAL GOLD SPONSOR • UNICOD CONSORTIUM • ADOPT A CHARACTER

ADOPT A CHARACTER

Search ...

U+11BA U+0AAC U+05D4 U+30EE U+1F680 U+3064 U+0A6C U+1F603 U+2691 U+04E9

U+2660 U+261E U+0B18 U+FF20 U+2325 U+30CE U+2192 U+0695 U+0D26 U+0222

U+1F403 U+FF0D U+104D U+00D2 U+3011 U+76BF U+25CF U+A755 U+042E U+1F49A

Struttura del Calcolatore

# UNICODE

---

## Emoji List, v12.1

For the new beta version, see [v13.0.8](#).

[Index & Help](#) | [Images & Rights](#) | [Spec](#) | [Proposing Additions](#)

This chart provides a list of the Unicode emoji characters and sequences, with single image and annotations. Clicking on a Sample goes to the emoji in the [full list](#). The ordering of the emoji is based on the code point in the **Code** column. [Recently-added emoji](#) are marked by a in the name and outlined images.

Emoji with skin-tones are not listed here: see [Full Skin Tone List](#).

For counts of emoji, see [Emoji Counts](#).

While these charts use a particular version of the [Unicode Emoji data files](#), the images and format may be updated at any time. For any production usage, consult those data files. For info header. For further information, see [Index & Help](#).

Smileys & Emotion				
face-smiling				
Nº	Code	Sample	CLDR Short Name	Other Keywords
1	<a href="#">U+1F600</a>		grinning face	face   grin   grinning face
2	<a href="#">U+1F603</a>		grinning face with big eyes	face   grinning face with big eyes   mouth   open   smile
3	<a href="#">U+1F604</a>		grinning face with smiling eyes	eye   face   grinning face with smiling eyes   mouth   open   smile
4	<a href="#">U+1F601</a>		beaming face with smiling eyes	beaming face with smiling eyes   eye   face   grin   smile
5	<a href="#">U+1F606</a>		grinning squinting face	face   grinning squinting face   laugh   mouth   satisfied   smile
6	<a href="#">U+1F605</a>		grinning face with sweat	cold   face   grinning face with sweat   open   smile   sweat
7	<a href="#">U+1F923</a>		rolling on the floor laughing	face   floor   laugh   rolling   rolling on the floor laughing
8	<a href="#">U+1F602</a>		face with tears of joy	face   face with tears of joy   joy   laugh   tear
9	<a href="#">U+1F642</a>		slightly smiling face	face   slightly smiling face   smile
10	<a href="#">U+1F643</a>		upside-down face	face   upside-down
11	<a href="#">U+1F609</a>		winking face	face   wink   winking face

# Codifiche di UNICODE

---

- Esistono diverse possibili rappresentazioni dei codepoint UNICODE. Una prima rappresentazione semplice, la UCS-2, usa 2 byte per rappresentare il valore di ciascun codepoint (limitata alla versione UNICODE 1.0 su 16 bit).
- In questo caso occorre specificare se la rappresentazione segue l'ordinamento Little Endian o Big Endian
  - Il problema dell'ordinamento dei byte in memoria naturalmente non si pone per i tipi di dato che possono essere codificati sul singolo byte come per esempio i caratteri che compongono un testo rappresentati secondo il codice ASCII

# Codifiche di UNICODE

---

- UTF-8 è un'altra possibile codifica concreta per i codepoint UNICODE a lunghezza variabile: la rappresentazione dei primi 127 codepoint di UNICODE in UTF-8 coincidono con la rappresentazione del codice ASCII base, i successivi 128 corrispondono alla prima estensione dell'ASCII (conosciuto col nome di Latin-1)
  - Codificare un testo ASCII in UCS-2 fa sprecare un sacco di spazio (raddoppia, dato che ASCII occupa 8 bit, mentre UCS-2 ne occupa 16)
- UTF-8: 8 bit UCS Transformation Format (dove UCS sta per Universal Character Set) è il default in diversi contesti (es. stringhe in Java, standard per il web e e-mail richiedono che i browser o client di posta supportino almeno UTF-8).

# UTF-8

---

- Codifica a lunghezza variabile: da 1 a 4 byte
- Il primo formato è per l'ASCII base, per cui la codifica risulta identica all'ASCII se si usano solo i 127 caratteri base
- Tutti i byte successivi al primo iniziano con 10

Bit	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	0ddddddd					
11	110ddddd	10dddddd				
16	1110dddd	10dddddd	10dddddd			
21	11110ddd	10dddddd	10dddddd	10dddddd		
26	111110dd	10dddddd	10dddddd	10dddddd	10dddddd	
31	1111110x	10dddddd	10dddddd	10dddddd	10dddddd	10dddddd

**Figura 2.45** Lo schema di codifica UTF-8.  
Struttura del Calcolatore

# Caratteristiche di UTF-8

---

- Lunghezza variabile, più difficile individuare dove inizia e termina ogni carattere ( - )
- Prefissi diversi per primo byte e byte successivi ( + ): non si rischia di perdere l'allineamento
- Non è legato all'ordine dei byte (big endian vs little endian) ( + )
- Compatibile con ASCII (su 7 bit) ( + )
- Risparmia spazio rispetto a quelli a lunghezza fissa ( + )

# UTF-16

---

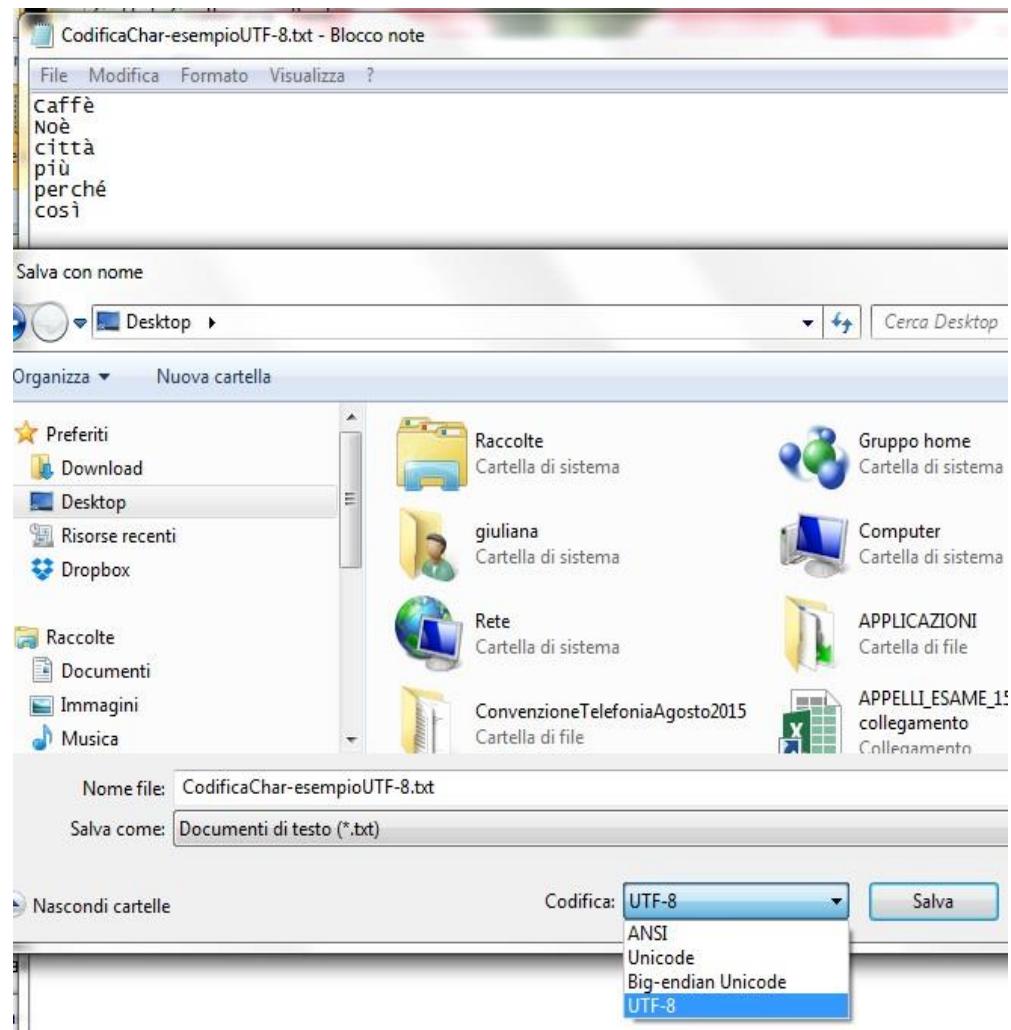
- I codepoint su 16 bit, quindi da 0000 a FFFF sono rappresentati come in UCS-2
- I codepoint da 20 bit sono scomposti in due parole da 16 bit: la prima è D800 or 10 bit più significativi del codepoint, la seconda è DC00 or 10 bit meno significativi del codepoint

# Esempio di codifiche diverse

Il Notepad di Windows permette in fase di salvataggio di selezionare una fra 4 diverse codifiche:

- ANSI
- Unicode
- Big endian Unicode
- UTF-8

Il contenuto del file cambia in ciascun caso ...

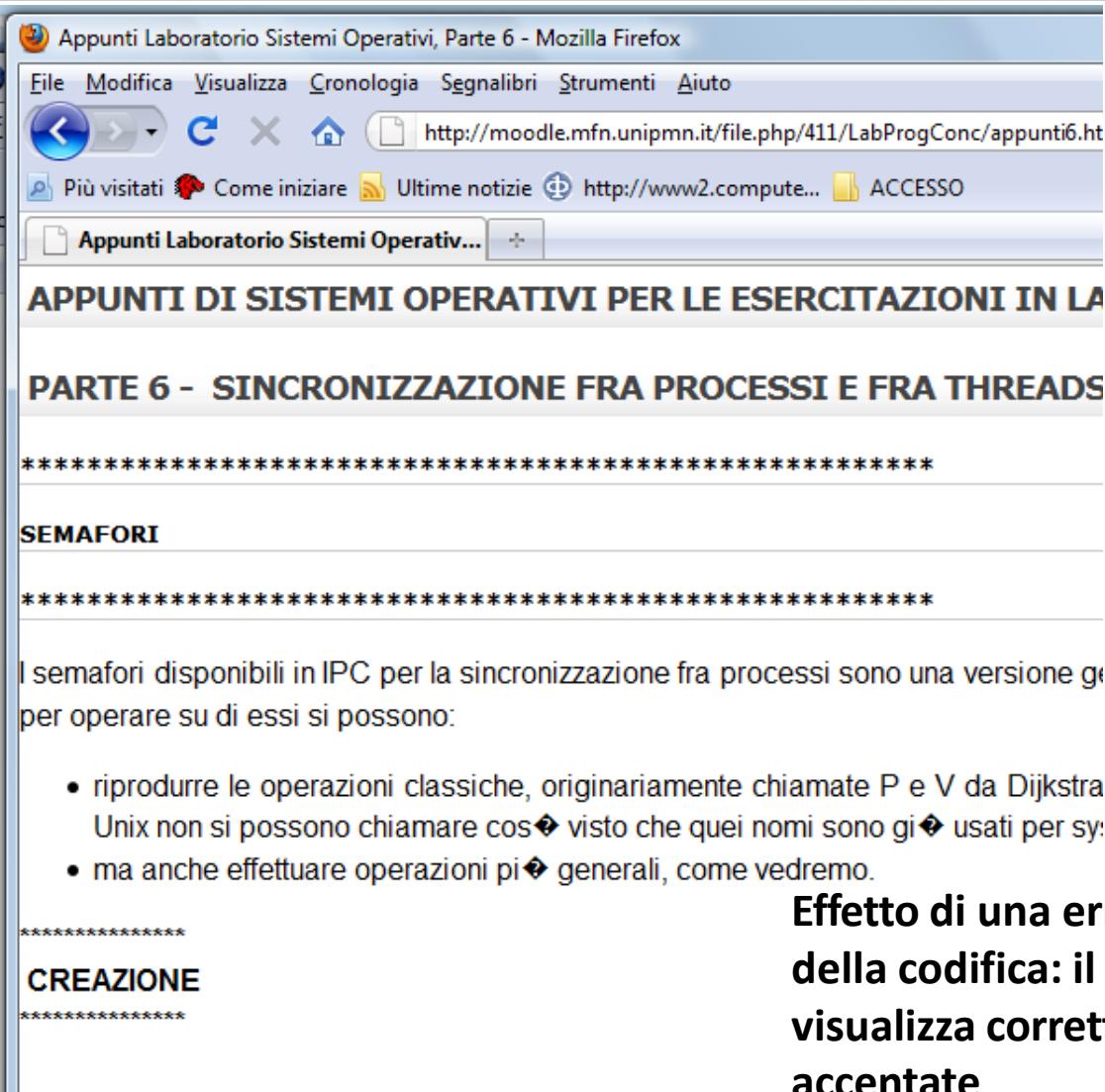


# Little Endian vs. Big Endian in UCS-2

---

- Un prefisso di 16 bit chiamato BOM (Byte Order Mark) indica qual è l'ordinamento usato:
- FFFE – Little Endian FEFF – Big Endian

# Errori di Codifica



The screenshot shows a Mozilla Firefox window with the title "Appunti Laboratorio Sistemi Operativi, Parte 6 - Mozilla Firefox". The address bar displays the URL <http://moodle.mfn.unipmn.it/file.php/411/LabProgConc/appunti6.htm>. The page content is titled "APPUNTI DI SISTEMI OPERATIVI PER LE ESERCITAZIONI IN LABORATORIO" and "PARTE 6 - SINCRONIZZAZIONE FRA PROCESSI E FRA THREADS". The text discusses semaphores and their operations. A section on "CREAZIONE" is partially visible at the bottom. The text contains several encoding errors, particularly with the letter 'ç'.

\*\*\*\*\*

**SEMAFORI**

\*\*\*\*\*

I semafori disponibili in IPC per la sincronizzazione fra processi sono una versione generalizzata dei semafori di Petri. Per operare su di essi si possono:

- riprodurre le operazioni classiche, originariamente chiamate P e V da Dijkstra
- ma anche effettuare operazioni più generali, come vedremo.

\*\*\*\*\*

**CREAZIONE**

\*\*\*\*\*

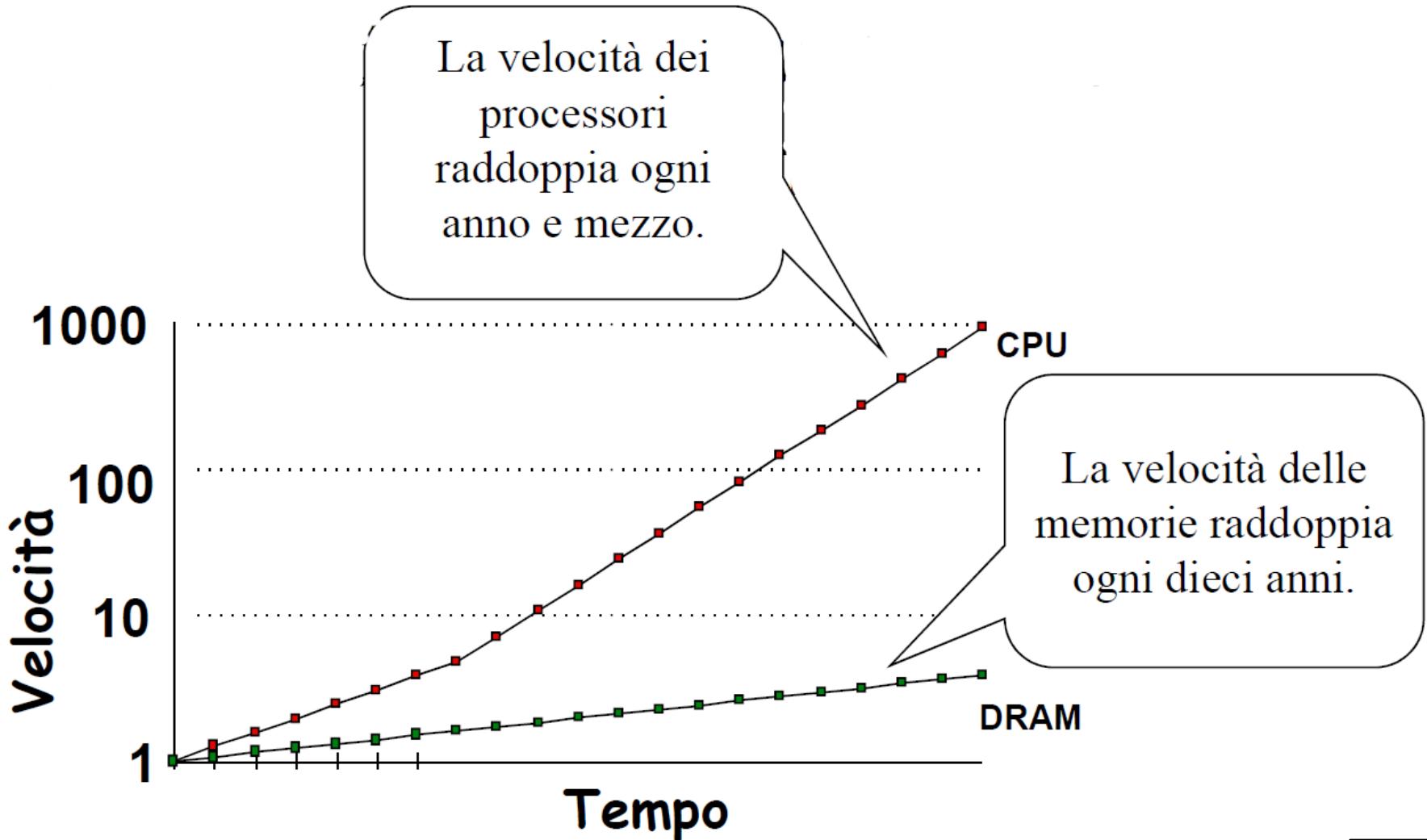
**Effetto di una errata interpretazione della codifica: il browser non visualizza correttamente le lettere accentate**

# Introduzione alle memorie

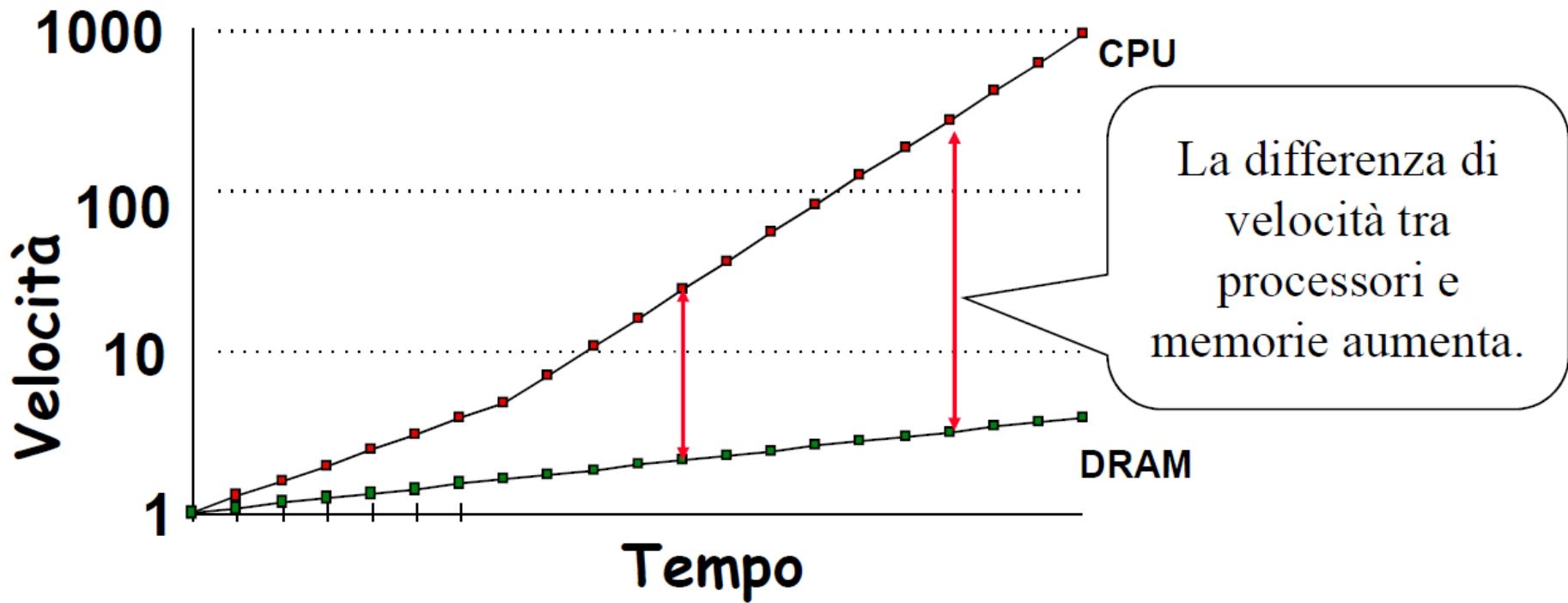
---

- Ogni sistema di elaborazione contiene dispositivi per la memorizzazione di dati ed istruzioni.
- L'insieme di tali dispositivi, e degli algoritmi per la loro gestione, costituisce il **sotto-sistema di memoria**.
- Tale sotto-sistema deve essere realizzato in modo che
  - il processore debba attendere il meno possibile per accedere a dati o istruzioni
  - il costo del sistema di memoria sia minimo (il costo per bit delle memorie è proporzionale alla loro velocità).
- Si deve quindi cercare un compromesso tra il costo del sotto-sistema di memoria e le sue prestazioni.

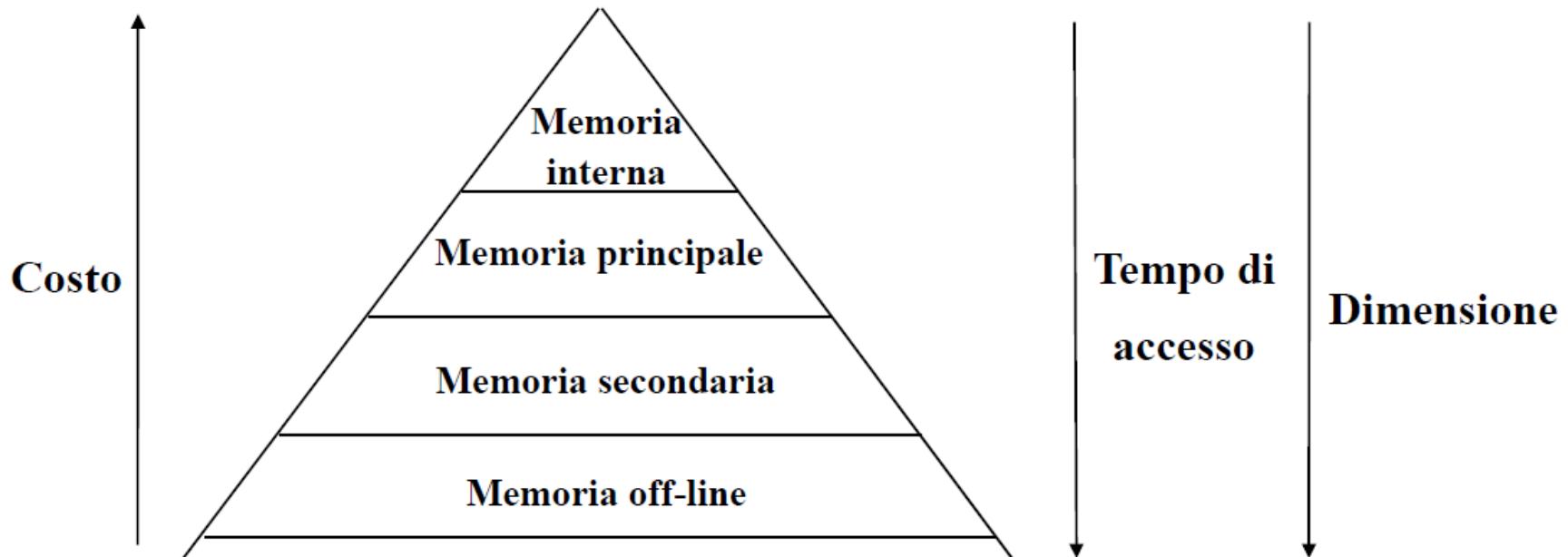
# Velocità del processore e della memoria



# Velocità del processore e della memoria



# Gerarchia delle memorie

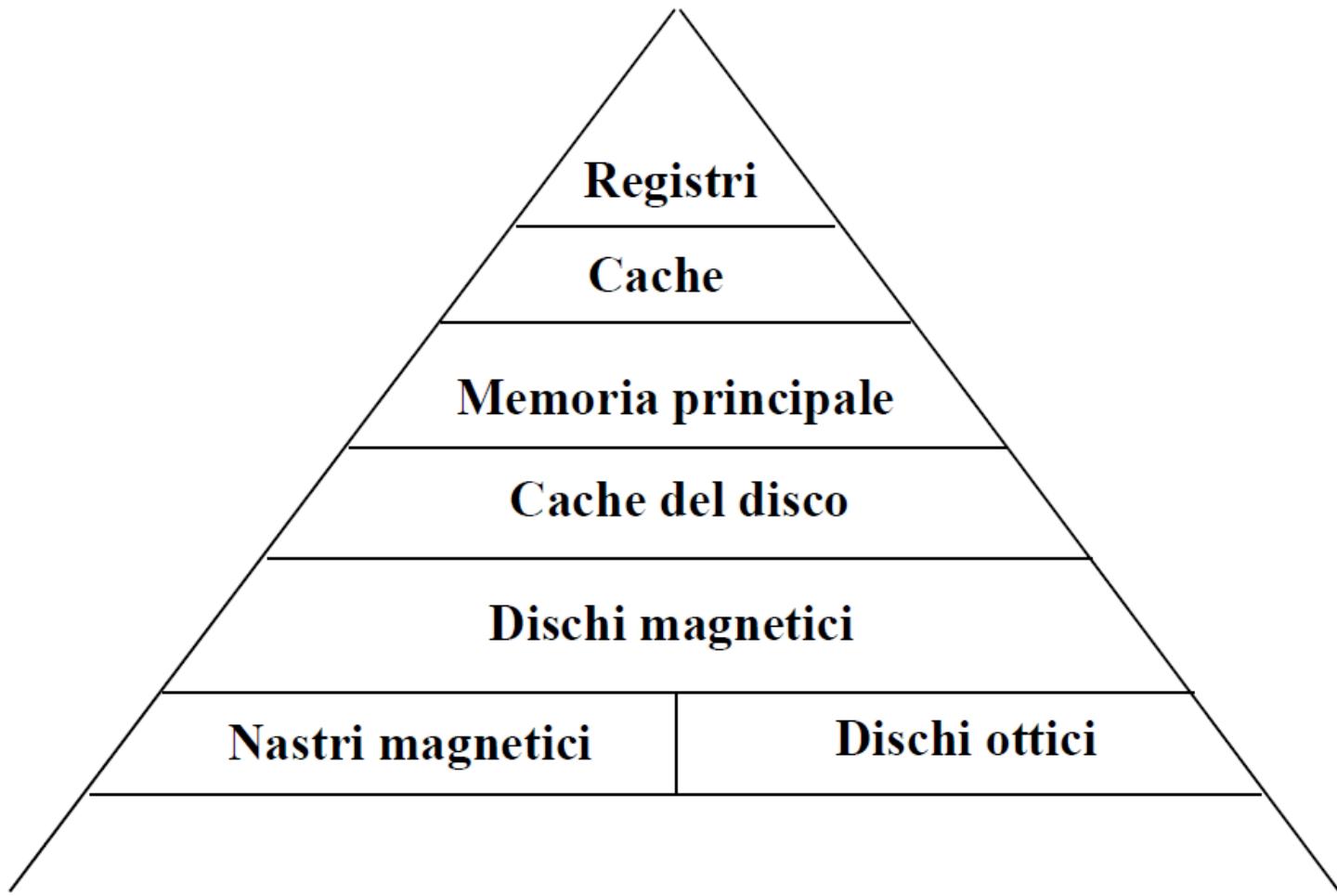


I tempi di accesso crescono  
verso il basso:  
Registri:  $\leq 1$  nsec (periodo clock)  
Cache: pochi nsec RAM:  
decine di nsec SSD: centinaia  
di nsec Dischi magnetici: msec

La capacità cresce verso il basso:  
Registri: decine di byte Cache: decine  
KB – qualche MB  
RAM: alcuni GB SSD: centinaia di GB  
Dischi magnetici: TB Nastri, dischi  
ottici: supporti rimovibili –  
capacità potenzialmente  
infinita

# Situazione attuale

---



# Memoria interna alla CPU

---

Corrisponde ai registri interni della CPU. È caratterizzata da:

- alta velocità (comparabile con quella del processore): i tempi di accesso sono dell'ordine dei ns
- limitate dimensioni (al più qualche Kbyte).

È di solito realizzata tramite celle di *RAM statica*.

# Memoria principale

---

Ha dimensioni molto maggiori (da qualche Mbyte fino a qualche Gbyte) ma tempi di accesso più elevati, dell'ordine delle decine di ns.

È accessibile in modo diretto tramite indirizzi.

È normalmente realizzata sotto forma di circuito integrato. È di solito realizzata tramite circuiti di *RAM dinamica*.

# Memoria secondaria

---

Ha dimensioni e tempi di accesso ancora maggiori, dell'ordine delle decine di ms.

Ha dimensioni che possono arrivare ai Tbyte.

Viene usata per memorizzare dati e programmi non immediatamente richiesti dal processore.

L'accesso è gestito da appositi programmi di interfaccia.

È normalmente realizzata sotto forma di *dischi magnetici* o memorie *Flash* (o combinazioni dei due).

Le informazioni memorizzate nella memoria secondaria non vengono perse allo spegnimento del sistema.

# Memoria Off-line

---

Permette di memorizzare grandi moli di dato (Pbyte) con tempi di accesso elevati (decine di secondi).

In taluni casi l'accesso alla memoria off-line richiede l'intervento di un operatore.

Di solito è composta da *dischi ottici* o *nastri*.

# Cache

---

Le **cache** sono memorie estremamente veloci che si interpongono tra il processore e la memoria principale.

All'interno di una cache risiedono temporaneamente i dati/programmi utilizzati in quel momento.

Il loro uso è trasparente al programmatore e al Sistema Operativo.

Le cache permettono di aumentare la velocità di accesso ai dati nella memoria principale senza ricorrere per essa a memorie di tipo più costoso.

# Strategia generale

---

Nella progettazione di un sistema di memoria vengono tenuti in conto i seguenti punti:

- conviene che il sistema complessivo sia composto da memorie di tipo e costo diversi, rispondenti ai diversi usi che della memoria vengono fatti (*gerarchia di memoria*)
- la gestione della memoria deve essere il più possibile trasparente per il programmatore e l'utente (*memoria virtuale*)
- se il sistema è di tipo *multiprocessore*, ogni processore deve poter lavorare con la memoria al massimo della velocità e senza interferire con il lavoro degli altri.

# Classificazione delle tecnologie di memoria

---

- È basata su vari parametri, alcuni sono:
  - costo
  - velocità
  - modi di accesso
  - alterabilità
  - affidabilità
  - caratteristiche fisiche.

# Costo

---

Comprende anche il costo della circuiteria ed eventualmente del software per la gestione delle interfacce necessarie all'uso della memoria.

È normalmente misurato in dollari/bit o \$/Mbyte.

# Velocità

---

Si esprime attraverso tre parametri

- il tempo di accesso
- il tempo di ciclo
- il tasso di trasferimento.

# Tempo di accesso (o latenza)

---

È il tempo che intercorre tra l'istante in cui all'unità di memoria giunge la richiesta di eseguire un'operazione (lettura o scrittura), e quello in cui tale operazione è eseguita.

È possibile che il tempo di accesso *in lettura* differisca da quello *in scrittura*.

Il tempo di accesso è di solito inversamente proporzionale al costo della memoria.

# Tempo di ciclo

---

È il tempo che deve intercorrere tra l'inizio di un ciclo di accesso alla memoria e l'inizio del ciclo successivo.

È chiaramente maggiore o uguale del tempo di accesso.

# Tasso di trasferimento

---

È la velocità con la quale i dati possono esser trasferiti verso o dall'unità di memoria.

Per le memorie ad accesso casuale è l'inverso del tempo di ciclo.

Per le altre vale che

$$T_N = T_A + n/R$$

dove

- $T_N$  è il tempo medio per leggere o scrivere  $n$  bit
- $T_A$  è il tempo medio di accesso
- $n$  è il numero di bit
- $R$  è il tasso di trasferimento (in bit per secondo, o *bps*).

# Evoluzione delle memorie

---

L'evoluzione della tecnologia presenta 2 tendenze:

- riduzione del *costo* per bit
- riduzione (meno marcata) del *tempo di accesso*.

# Modi di accesso

---

Sono principalmente:

- *sequenziale*: le informazioni possono essere lette/scritte solo in un ordine prefissato; è il caso dei nastri
- *diretto*: ogni blocco ha un indirizzo, e la ricerca nel blocco è sequenziale; è il caso dei dischi magnetici
- *casuale*: ogni unità di dato ha un indirizzo, e le informazioni possono essere lette/scritte in qualsiasi ordine; il tempo di accesso è uguale per tutte le locazioni di memoria; è il caso delle memorie a semiconduttore
- *associativo*: l'accesso avviene tramite un confronto tra il contenuto di ogni cella e quello specificato in una maschera; è il caso di taluni tipi di cache.

# Alterabilità

---

Vi sono memorie il cui contenuto può essere scritto una volta sola (*Read Only Memory o ROM*).

In alcuni casi il contenuto di una memoria ROM può essere modificato off-line (*Programmable ROM o PROM*).

# Affidabilità

---

Le memorie possono essere colpite da due tipi di guasto:

- *Guasti transitori*: uno o più bit cambiano valore ad un certo istante, ma la memoria continua a funzionare correttamente
- *Guasti permanenti*: qualcosa nella memoria smette definitivamente di funzionare.

L'affidabilità è di solito misurata attraverso i seguenti parametri:

- *Mean Time To Failure* (MTTF): tempo medio prima di avere un guasto permanente
- *Failure rate* (inverso del *Mean Time Between Failures* o MTBF): frequenza media di occorrenza dei guasti transitori.

# Altre caratteristiche

---

- Tipo della memoria (elettronica, magnetica, meccanica, ottica)
- Consumo
  - può comportare la necessità di sistemi di raffreddamento
  - può essere critico per i sistemi portabili
- Portabilità
- Robustezza (ad esempio rispetto alle sollecitazioni meccaniche)
- Dimensione: dipende dalla densità di immagazzinamento.

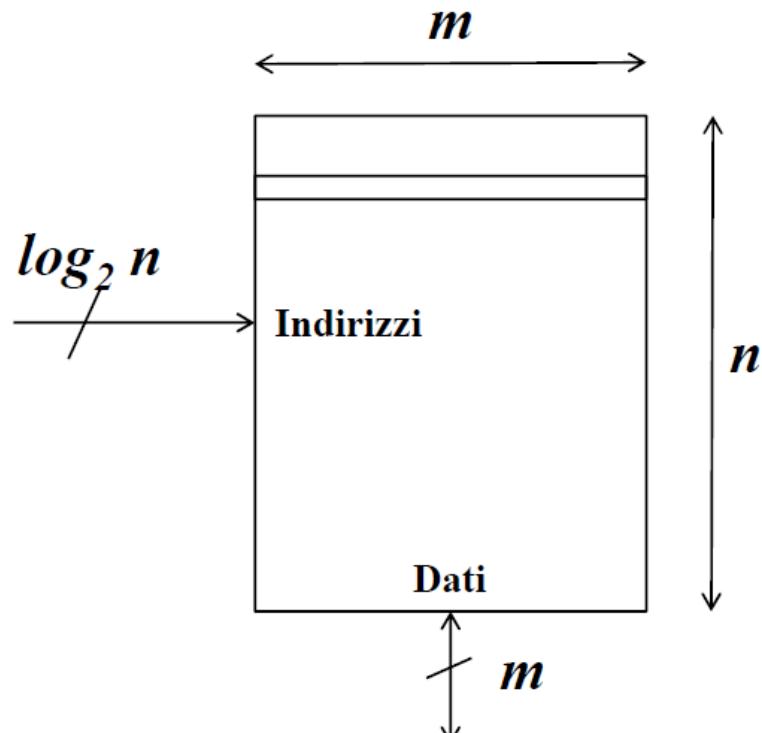
# Le memorie ad accesso casuale

---

- Ogni cella può essere indirizzata indipendentemente
- I tempi di accesso sono uguali e costanti per ogni cella
- Lettura:
  - Sull'Abus si scrive l'indirizzo della cella da leggere
  - Sul Cbus si attivano i segnali di lettura
  - Sul Dbus la memoria scrive il contenuto della cella.
- Scrittura:
  - Sull'Abus si mette l'indirizzo della cella da leggere
  - Sul Cbus si attivano i segnali di scrittura
  - Sul Dbus si scrive il valore da memorizzare nella cella.

# Le memorie ad accesso casuale

- Sono caratterizzate da
  - numero di parole ( $n$ )
  - numero di bit per parola ( $m$ ).
- Il numero di segnali di ingresso/uscita conseguentemente
  - $\log_2 n$  per i segnali di indirizzo
  - $m$  per i segnali di dato.



# Le memorie ad accesso casuale

---

- Classificazione
  - ROM (*Read Only Memory*)
  - PROM (*Programmable Read Only Memory*)
  - EPROM (*Electrically Programmable Read Only Memory*)
  - EEPROM (*Electrically Erasable Programmable Read Only Memory*)
  - Flash
  - RAM

# ROM

---

- ❑ Applicazioni:
  - ❑ librerie di procedure frequentemente usate
  - ❑ programmi di sistema
  - ❑ tavole di funzioni.
- ❑ La definizione del contenuto avviene prima della realizzazione del silicio, ed il contenuto non è in alcun modo modificabile in seguito.
- ❑ Il costo è dominato dal costo fisso per allestire la linea di produzione del dispositivo, ed è quasi indipendente dal numero di chip prodotti.
- ❑ L'impatto di qualsiasi errore di progetto è drammatico.

# PROM

---

- La scrittura è eseguita a valle del processo di produzione tramite speciali attrezzature denominate *programmatori*.
- La scrittura può avvenire una volta sola.
- La PROM è non volatile.
- Sono preferibili alle ROM per bassi volumi.

# EPROM

---

- ❑ Possono essere riprogrammate, previa precedente cancellazione tramite esposizione prolungata (~20 min) a luce ultravioletta.
- ❑ La scrittura può avvenire un numero indefinito di volte.
- ❑ La scrittura può avvenire anche dopo il montaggio sulla scheda.
- ❑ Sono più costose delle PROM.

# EEPROM

---

- Le EEPROM (o E<sup>2</sup>PROM) possono essere riprogrammate byte per byte anche dopo il montaggio sulla scheda, ma l'operazione richiede più tempo di quella di lettura (centinaia di ms).
- La scrittura può essere eseguita tramite i normali canali (bus) e segnali.
- Sono più costose delle EPROM.

# Flash

---

- ❑ Il costo è intermedio tra quello di EPROM ed EEPROM.
- ❑ In lettura si comportano come le RAM, mentre le operazioni di scrittura
  - ❑ sono più lente
  - ❑ vanno eseguite a blocchi
  - ❑ richiedono una precedente operazione di cancellazione

# RAM

---

Sono di 2 tipi:

- ❑ memorie statiche (o SRAM):
  - ❑ la singola cella corrisponde a un flip flop
- ❑ memorie dinamiche (o DRAM):
  - ❑ la singola cella corrisponde a un condensatore e a un transistor
  - ❑ l'informazione è memorizzata sotto forma di carica del condensatore
  - ❑ richiedono un rinfresco periodico dell'informazione
  - ❑ la lettura è di tipo distruttivo (Destructive Read-Out).

# RAM

---

- Per leggere o scrivere una parola si deve attivare la relativa linea di parola.
- Quando la linea di parola non è attivata, il relativo flip flop/ condensatore mantiene il proprio valore.
- Quando la linea di parola è attivata, è possibile
  - leggere i valori (opposti) forzati dal flip flop sulle due linee di dato (lettura)
  - scrivere un nuovo valore, forzando due valori opposti sulle linee di dato (scrittura)

# RAM

---

- Le RAM statiche sono (rispetto a quelle dinamiche):
  - più veloci
  - più costose (in termini di area di silicio richiesta, quindi meno dense)
  - più semplici da utilizzare
  - più affidabili.

# SDRAM

---

- Le SDRAM –RAM Dinamiche Sincrone –un ibrido (parte statica, parte dinamica), guidata dal clock di sistema, non necessita di segnali di controllo per sapere quando deve rispondere.
- Ad ogni ciclo trasferisce 4, 8, o 16 bit

# Riassunto memorie ad accesso casuale

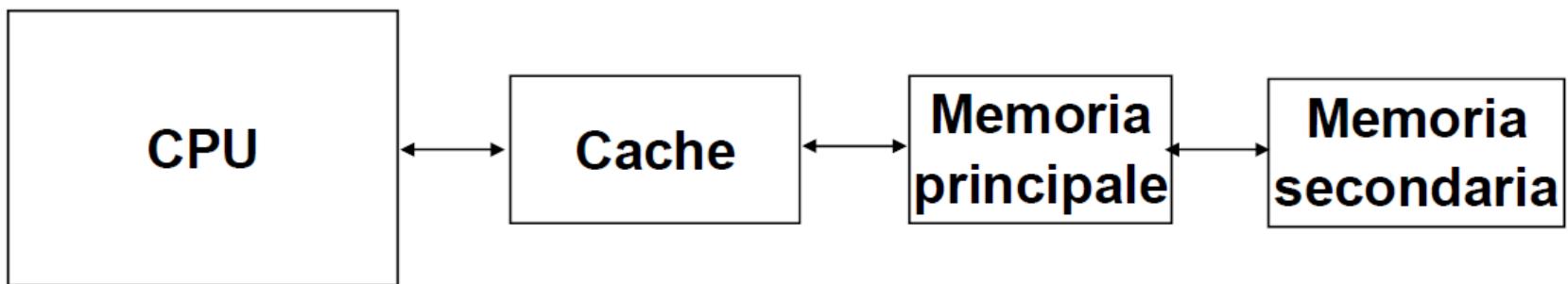
---

Tipo	Categoria	Cancellazione	Byte modificabili	Volatile	Tipico utilizzo
SRAM	Read/write	Elettrica	Sì	Sì	Cache di secondo livello
DRAM	Read/write	Elettrica	Sì	Sì	Memoria centrale (vecchia)
SDRAM	Read/write	Elettrica	Sì	Sì	Memoria centrale (recente)
ROM	Read-only	Impossibile	No	No	Elettrodomestici (prodotti in grandi volumi)
PROM	Read-only	Impossibile	No	No	Dispositivi (prodotti in piccoli volumi)
EPROM	Read-mostly	Raggi UV	No	No	Prototipazione di dispositivi
EEPROM	Read-mostly	Elettrica	Sì	No	Prototipazione di dispositivi
Flash	Read/write	Elettrica	No	No	“Pellicola” per macchine fotografiche digitali

# Le memoria cache

---

**Le memorie cache sono memorie di piccole dimensioni ma con elevata velocità interposte tra il processore e la memoria principale.**



# Località dei riferimenti

---

**La presenza di una cache può migliorare le prestazioni di un sistema per via della *località dei riferimenti* osservabile nella maggioranza dei programmi.**

**Si esprime in due forme:**

- **località *temporale*:** se all'istante  $t$  il programma fa accesso ad una cella di memoria, è molto probabile che il programma faccia nuovamente accesso alla stessa cella entro l'istante  $t + D$
- **località *spaziale*:** se all'istante  $t$  il programma fa accesso alla cella di memoria di indirizzo  $X$ , è molto probabile che entro l'istante  $t + D$  il programma faccia accesso anche alla cella di indirizzo  $X + e$ .

# Principio di funzionamento

---

**Se all'istante  $t$  (primo accesso ad un blocco di memoria da parte del programma) viene caricato nella cache l'intero blocco, ci sono alte probabilità che per un certo tempo  $D$  il programma trovi nella cache tutte le parole di memoria cui deve fare accesso.**

# Prestazioni

---

**Si definiscano le seguenti grandezze:**

- $h$ : hit ratio della cache
- $C$ : tempo di accesso alla cache
- $M$ : tempo di accesso in memoria quando il dato non è in cache.

**Il tempo medio di accesso in memoria per la CPU sarà**

$$t_{\text{medio}} = hC + (1-h)M$$

**Valori normali per  $h$  sono dell'ordine di 0,9.**

# Funzionamento della cache

---

- La cache si interpone tra processore e memoria principale.
- Ogni volta che il processore esegue un accesso alla memoria la cache (1) intercetta l'indirizzo; (2) verifica se il blocco cui appartiene la parola è presente nella cache,
  - se sì: estrae la parola dal blocco e la fornisce alla CPU al posto (e prima) della memoria principale (*hit*)
  - se no: provvede a caricare nella cache l'intero blocco di cui la parola fa parte (*miss*).

# Funzionamento della cache

---

**In caso di hit, la cache riduce i tempi di accesso di un fattore tra 2 e 6 (indicativamente).**

**In caso di miss, la cache risponde in due possibili modi:**

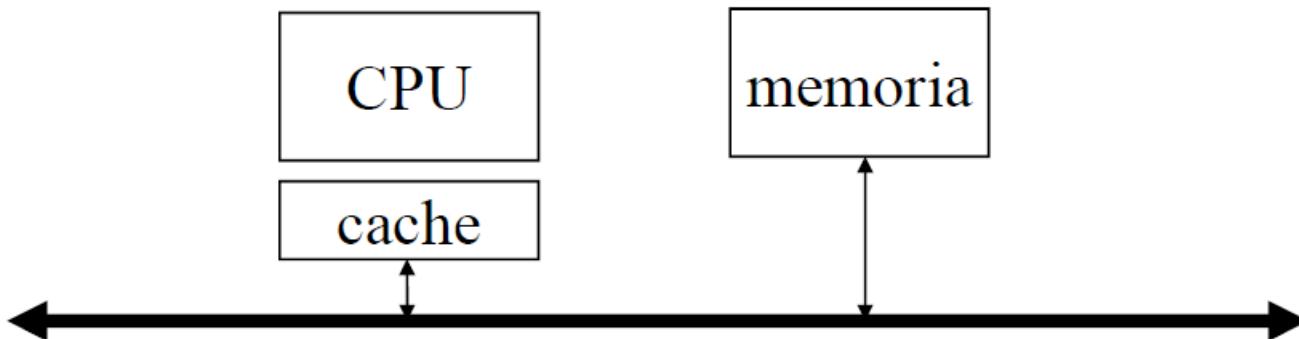
- **Accede alla memoria e carica l'intero blocco mancante; poi fornisce la parola richiesta. Il tempo di accesso è quindi superiore a quello di accesso alla memoria senza cache.**
- **Accede alla memoria e fornisce subito la parola richiesta; poi provvede al caricamento del blocco (*load-through* o *early restart*). Questa tecnica richiede un maggior costo dell'hardware della cache, ma il miss ha un impatto più limitato sulle prestazioni della cache.**

# Posizione della cache

**La cache è normalmente collocata tra la CPU e il bus, anziché tra la memoria principale e il bus.**

**I vantaggi che si ottengono in questo modo sono:**

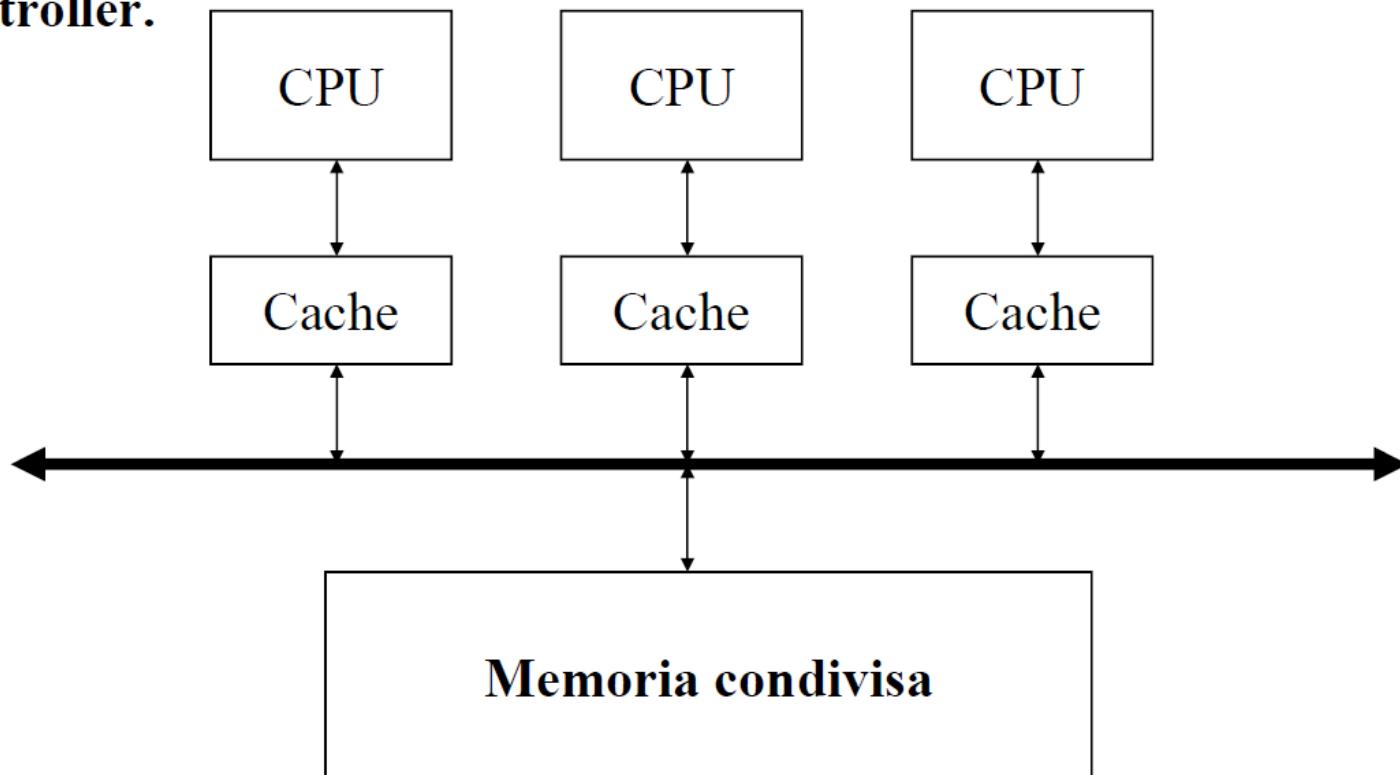
- **si alleggerisce il carico del bus**
- **la soluzione è compatibile con un'architettura multiprocessore.**



# Coerenza della cache

È un problema nei sistemi a multiprocessore con memoria condivisa, nei quali ogni processore ha una sua cache.

Problemi analoghi si possono avere se il sistema utilizza un DMA controller.



# Coerenza della cache

---

Per ottenere la coerenza delle cache si introduce per ogni linea di cache un *bit di validità*.

Se è disattivato, significa che il blocco presente in quella linea ha un valore diverso dal corrispondente blocco nella memoria principale. In tal caso ogni accesso al blocco comporta un miss.

# Memorie ad accesso seriale

---

**Le memorie ad accesso seriale (dischi, nastri, cassette) sono generalmente utilizzate per la memoria secondaria e off-line.**

**La memoria secondaria è normalmente composta da dischi magnetici e memorie flash; dischi ottici e nastri costituiscono normalmente la memoria off-line.**

**Le memorie ad accesso seriale sono caratterizzate da *basso costo ed elevato tempo di accesso*.**

# Memorie a disco magnetico

---

L'elemento di memoria è un disco ricoperto di materiale magnetico, su cui esistono una serie di *tracce* concentriche.

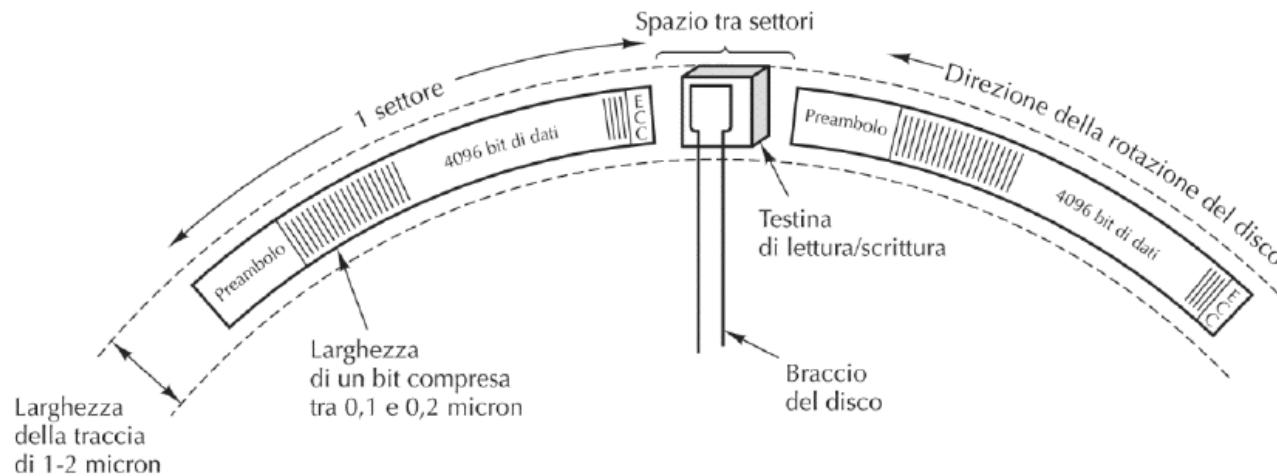
L'unità di memoria può essere costituita da più dischi: in tal caso essi sono connessi ad un unico asse e ruotano a velocità costante.

Ogni superficie (*faccia*) è dotata di una testina in grado di muoversi radialmente fin sulla traccia desiderata.

Le varie testine si muovono di solito in maniera solidale.

L'insieme delle tracce ad uguale distanza dal centro poste su facce diverse è denominato *cilindro*.

# Memorie a disco magnetico



**Figura 2.19** Porzione di una traccia del disco. Sono mostrati due settori.

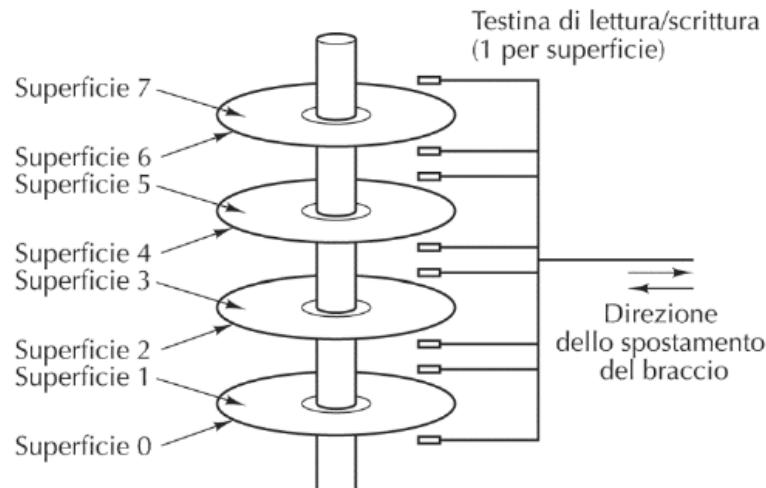
**Tempi di accesso:**

**Seek + mezza rotazione + trasferimento dati**

**5-10 msec +**

**3-6 msec + (rotazione 5400/7200/10800 rpm)**

**3.5  $\mu$ sec (tasso di trasferimento 150MB/sec)**



# Memorie a disco magnetico

---

- Per poter leggere/scrivere le testine devono posizionarsi sulla *traccia* giusta (effettuando un'operazione detta di seek)
- Successivamente si deve attendere che, ruotando, il settore d'interesse arrivi sotto la testina, dopo di che si trasferiscono i dati da leggere o scrivere

Nota: i settori potrebbero essere in numero maggiore sulle tracce più esterne, minori all'interno

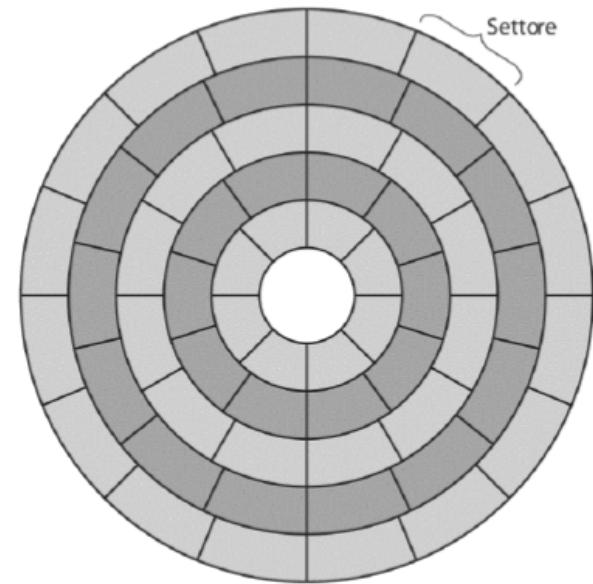


Figura 2.21 Disco con cinque zone. Ciascuna zona ha più tracce.

# Memorie a disco magnetico

---

Il tempo di accesso  $t_A$  è determinato da:

- $t_s$ : tempo per posizionare la testina sulla traccia opportuna (*seek time*)
- $t_L$ : tempo per posizionare la testina sul settore, all'interno della traccia (*latency time*)
- $t_D$ : tempo per leggere serialmente i dati (*data-transfer time*).

Si ha quindi che

$$t_A = t_s + t_L + t_D$$

# Modalità di indirizzamento dei dischi

---

Dal momento che  $t_s$  e  $t_L$  sono significativi rispetto a  $t_D$ , i dati sono normalmente raggruppati in *settori* (di dimensioni indicativamente pari a 1 KB), e la lettura/scrittura dei dati avviene utilizzando il settore come unità minima di accesso.

Detta  $T$  la densità (in bit/cm) di memorizzazione, e  $V$  la velocità (in cm/sec) della testina rispetto alla traccia, la velocità di trasferimento dati una volta che la testina è opportunamente posizionata è data da  $T \times V$ .

# Memorie a nastro magnetico

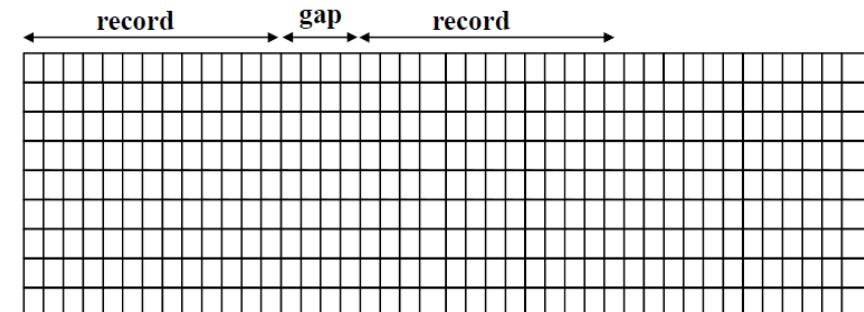
---

**In questo caso il supporto è un nastro di plastica flessibile largo 1/4 o 1/2 di pollice, su cui sono memorizzate 9 tracce parallele.**

**Ogni traccia possiede la sua testina, ed è quindi possibile leggere/scrivere contemporaneamente le 9 tracce.**

**I bit di ogni byte sono distribuiti tra le prime 8 tracce; l'ultima contiene un codice di parità.**

**I dati sono organizzati in record, che possono essere di dimensione fissa (e quindi sostituibili da altri) o variabile (ottimizzando l'uso del nastro).**



# Memorie ottiche

---

**Sono diffuse sotto forma di dischi ottici.**

**Hanno le seguenti caratteristiche:**

- **capacità intorno ai Gbyte**
- **elevati tempi di accesso (1s)**
- **data transfer rate come negli hard-disk (decine di Mbyte/s)**
- **elevata affidabilità (grazie alla mancanza di parti meccaniche vicine o a contatto, come nei dischi e nastri magnetici).**

# Tipologie di dischi ottici

---

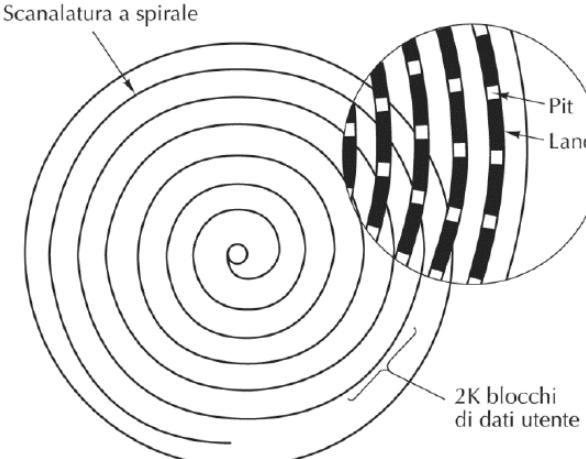
- **CD**
- **CD-ROM**
- **WORM**
- **CD-RW**
- **DVD**

**Differiscono per**

- **Capacità**
- **Alterabilità**
- **Meccanismo di funzionamento.**

# CD

- Sono dischi non cancellabili che memorizzano informazioni audio.
- Lo standard prevede dischi da 12 cm che possono memorizzare circa 60 minuti di registrazione sonora.
- La velocità di rotazione diminuisce a mano a mano che ci si allontana dal centro per garantire una velocità lineare costante: 530 RPM – 200 RPM
  - (2x, 4x, ..., 32x fattori moltiplicativi della velocità)
  - Velocità di rotazione molto distante da quella dei dischi magnetici (3600 RPM, 7200 RPM)



# CD-ROM

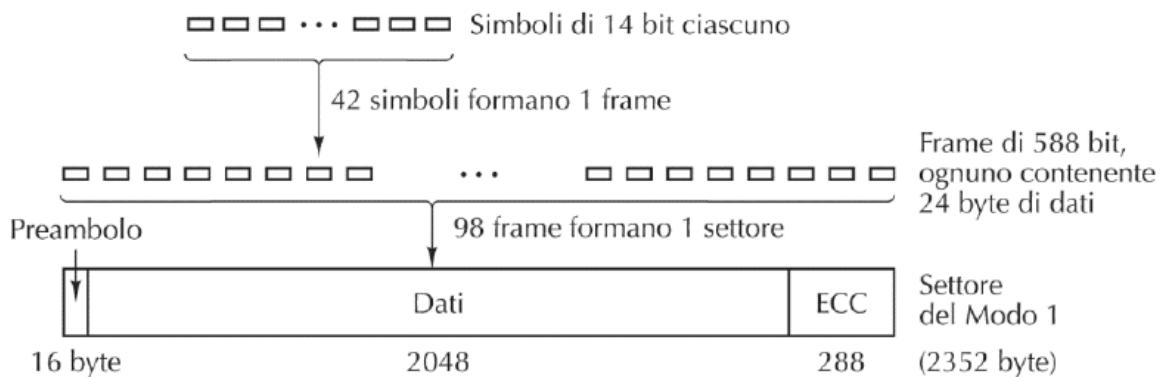


Figura 2.26 Struttura logica dei dati di un CD-ROM.

- In un CD-ROM le informazioni sono organizzate in settori con diversi livelli di ridondanza per garantire la possibilità di correggere errori. La quantità di dati utili rispetto al numero di bit memorizzati (per garantire diversi livelli di correzione di errori) è molto bassa: il 28% ( $2048 / 7203$ )
- A velocità singola vengono letti 75 settori/sec cioè 150Kbyte/sec. A velocità 32x

# CD-Rewritable (CD-RW)

---

**Furono lanciati da Philips e Sony nel 1997.**

**Possono essere riscritti un numero arbitrario di volte.**

**Il loro funzionamento si basa su una combinazione di argento, indio, antimonio e tellurio. Questa miscela ha le seguenti proprietà**

- quando viene riscaldata ad una certa temperatura ( $200^{\circ}$ ) e poi raffreddata, diventa cristallina**
- quando invece viene riscaldata ad una temperatura più elevata ( $500^{\circ}$ - $700^{\circ}$ ) e poi raffreddata, diventa amorfa**
- quando un raggio laser illumina una parte di disco resa cristallina, il sottostante strato metallico lo riflette; quando invece illumina una parte amorfa, viene assorbito.**

# Velocità di accesso

---

Per i driver in grado di scrivere un CD-R o CD-RW viene normalmente indicata da 3 numeri, che indicano (rispetto ad un'unità base pari a 150 KB/s)

- la velocità di scrittura
- la velocità di riscrittura
- la velocità di lettura.

## Esempio

**12x2x24**

# DVD (Digital Video Disk)

---

È il risultato di un accordo del 1996 tra 11 produttori.

Ha 4 formati:

- SD-5: single-sided/single-layered disc, storage capacity di 4.7 GB
- SD-9: single-sided/dual-layered disc, storage capacity di 8.5 GB
- SD-10: dual-sided/single-layered disc, storage capacity di 9.4 GB
- SD-18: a dual-sided/dual-layered disc, storage capacity di 17 GB.

Il tempo per leggere un'intera traccia è di 135 minuti.

È attualmente utilizzato principalmente per la memorizzazione di contenuti video.

# DVD e CD-ROM

---

	<b>DVD</b>	<b>CD-ROM</b>
<b>Diametro del disco</b>	<b>120 mm</b>	<b>120 mm</b>
<b>Spessore del disco</b>	<b>1,2 mm (0.6 mm x 2)</b>	<b>1,2 mm</b>
<b>Numero di superfici</b>	<b>1 o 2</b>	<b>1</b>
<b>Numero di strati</b>	<b>1 o 2</b>	<b>1</b>
<b>Diametro del buco centrale</b>	<b>15 mm</b>	<b>15 mm</b>
<b>Dimensione min. piazzole</b>	<b>0,4 µm</b>	<b>0,834 µm</b>
<b>Distanza tra le tracce</b>	<b>0,74 µm</b>	<b>1,6 µm</b>
<b>Velocità media di trasf.</b>	<b>4,7 Mbytes/sec</b>	<b>0,15 Mbytes/sec</b>
<b>Capacità (1 strato, 1 sup.)</b>	<b>5 Gbytes</b>	<b>0,682 Gbytes</b>
<b>Capacità (2 strati, 2 sup.)</b>	<b>17 Gbytes</b>	-

# Blue-ray Disk (BD)

---

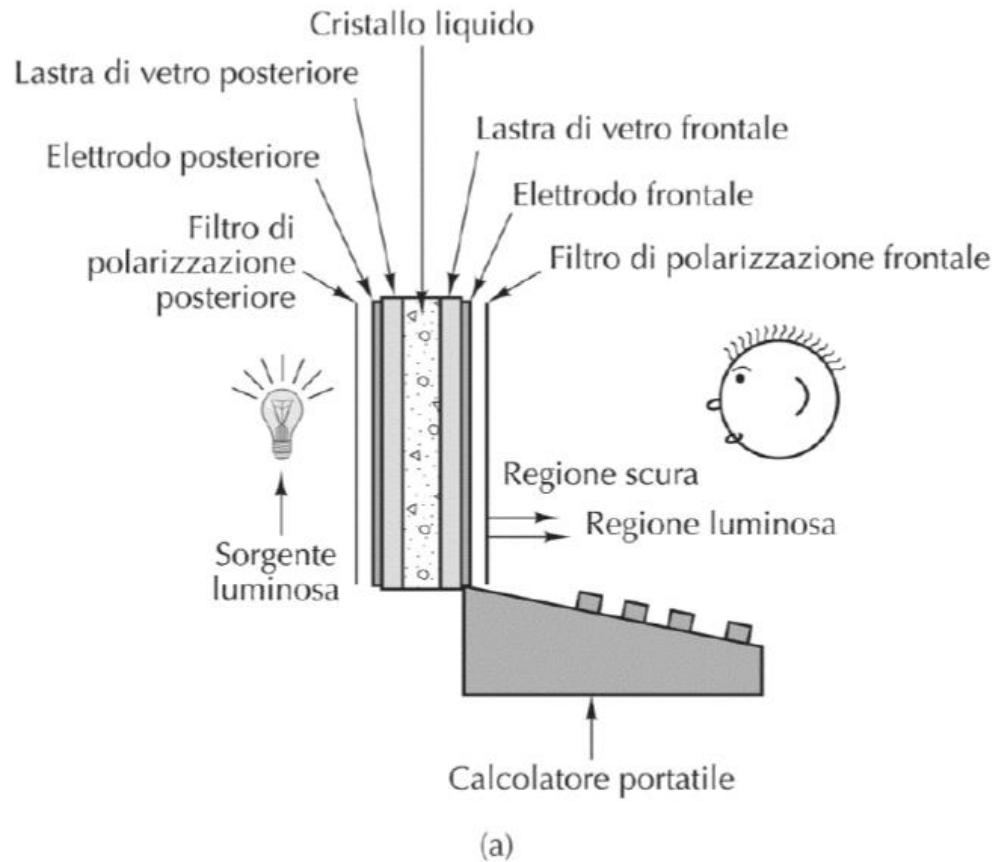
**Sono il formato utilizzato per memorizzare dati e video ad alta definizione.**

**Usa un raggio laser con lunghezza d'onda diversa da quella dei DVD (405 nm anzichè 650 nm).**

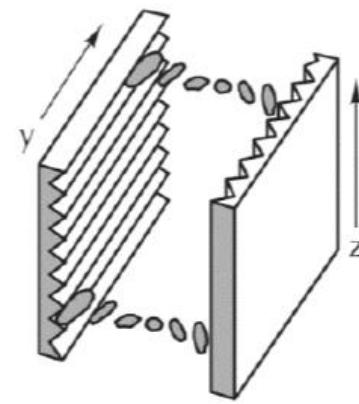
**Nella versione più capiente può memorizzare fino a 200 GB.**

I/O

# Schermo



(a)



(b)

**Figura 2.33** (a) Struttura di uno schermo LCD. (b) I solchi sulla lastra frontale e su quella posteriore sono perpendicolari tra loro.

# Schermo

---

Gli attuali schermi piatti utilizzano cristalli liquidi (LCD): le molecole di tali cristalli possono essere orientate tramite applicazione di campi elettrici, e in questo modo regolare il passaggio di luce in ciascun punto di una matrice (più o meno fitta a seconda della risoluzione, cioè del numero di **PIXEL** – Picture Elements distinti disponibili nella matrice).

# Le immagini sono formate da molti PIXEL

---

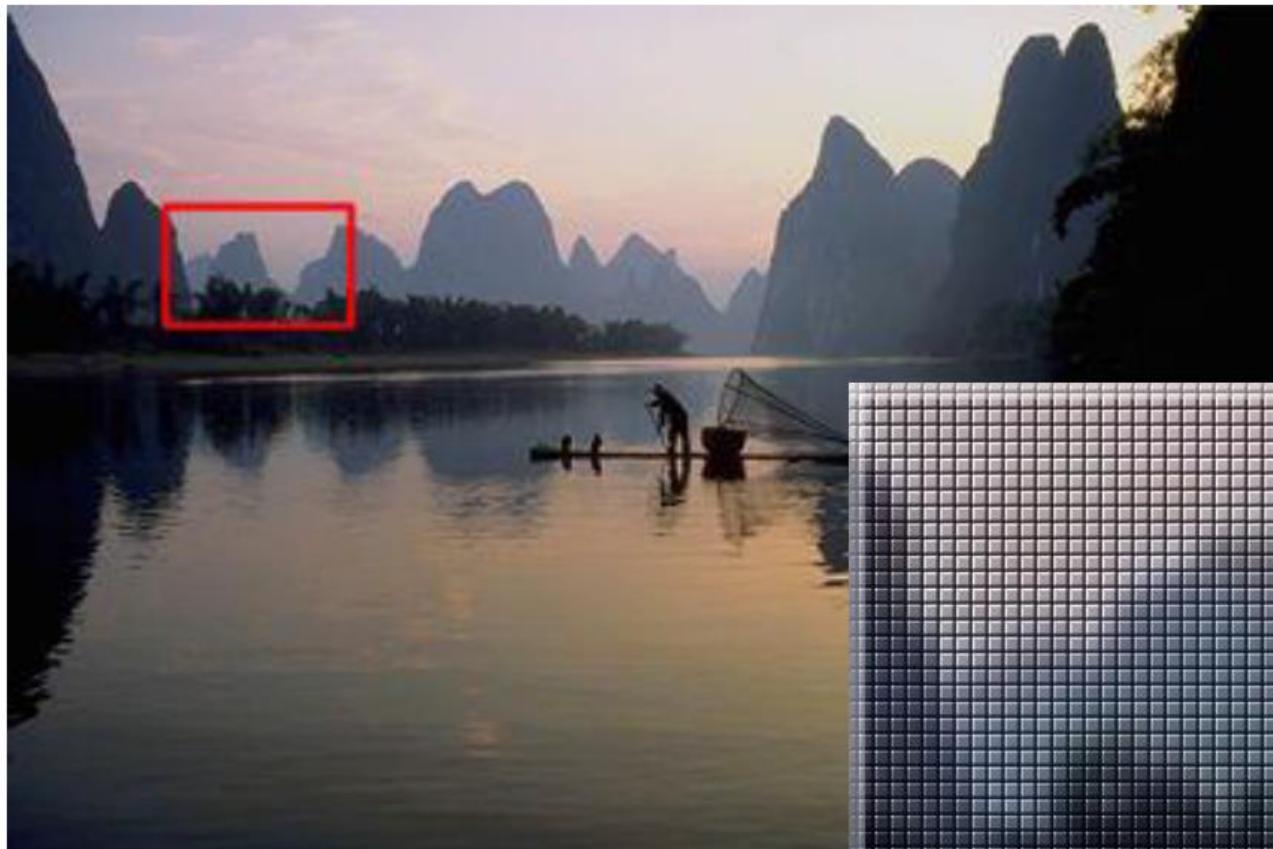
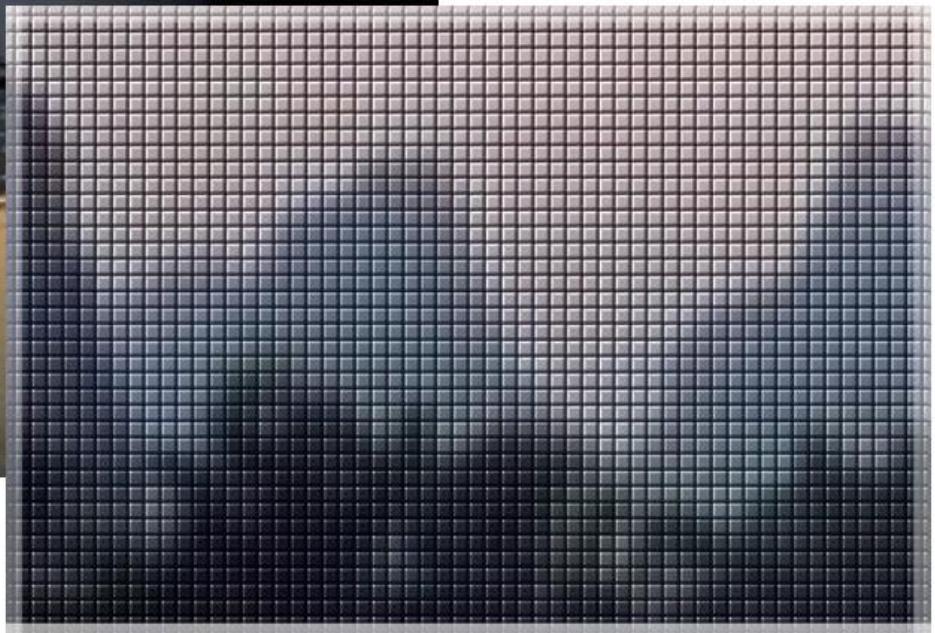


Immagine RASTER



# Rappresentazione Vettoriale vs Raster

---

La rappresentazione delle immagini come matrice di punti è chiamata «raster» (o anche bitmap). Una immagine può anche essere descritta (codificata e memorizzata) in modo diverso: la rappresentazione vettoriale descrive l'immagine come composizione di forme geometriche elementari posizionate in base a coordinate sul piano. Questa rappresentazione deve poi essere trasformata in mappa di pixel per essere visualizzata sullo schermo.

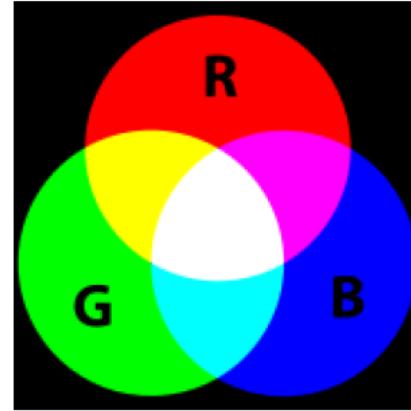
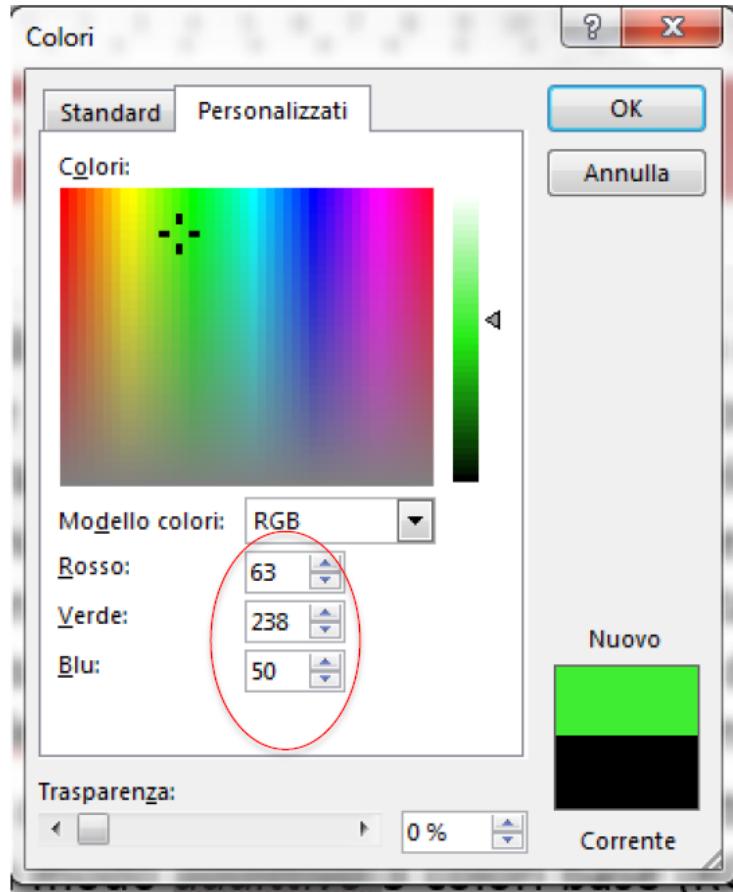
# Schermo

---

Ad ogni **PIXEL** occorre associare un colore.

Il colore di ciascun punto sullo schermo viene ottenuto «miscelando» in modo *additivo* 3 colori base (RGB, Red-Green-Blue). Utilizzando un byte per l'intensità di ogni colore (quindi 3 byte =24 bit in tutto) si possono avere fino a 16 milioni ( $=2^{24}$ ) di colori distinti.

# Composizione addittiva (RGB)



**Miscelazione addittiva dei colori  
Rosso, Verde, Blu**

**Servono 3 byte per pixel; in  
alternativa se ne possono usare  
di meno, es. 1 byte, ricorrendo  
ad una tavolozza di colori che ne  
contiene 255.**

# Risoluzione e profondità

---

- La **risoluzione** dell'immagine è il numero di pixel che la costituiscono, espressi in termini di larghezza x altezza. Ovviamente, aumentando il numero di pixel a disposizione, migliora la qualità dell'immagine.
- La **profondità** dell'immagine è invece il numero di bit che servono per rappresentare un singolo pixel dell'immagine.
- Il numero di bit richiesti per memorizzare un'immagine dipende dalla risoluzione e dalla profondità

numero di bit per immagine = risoluzione x profondità

# Diverse risoluzioni e spazio necessario

---

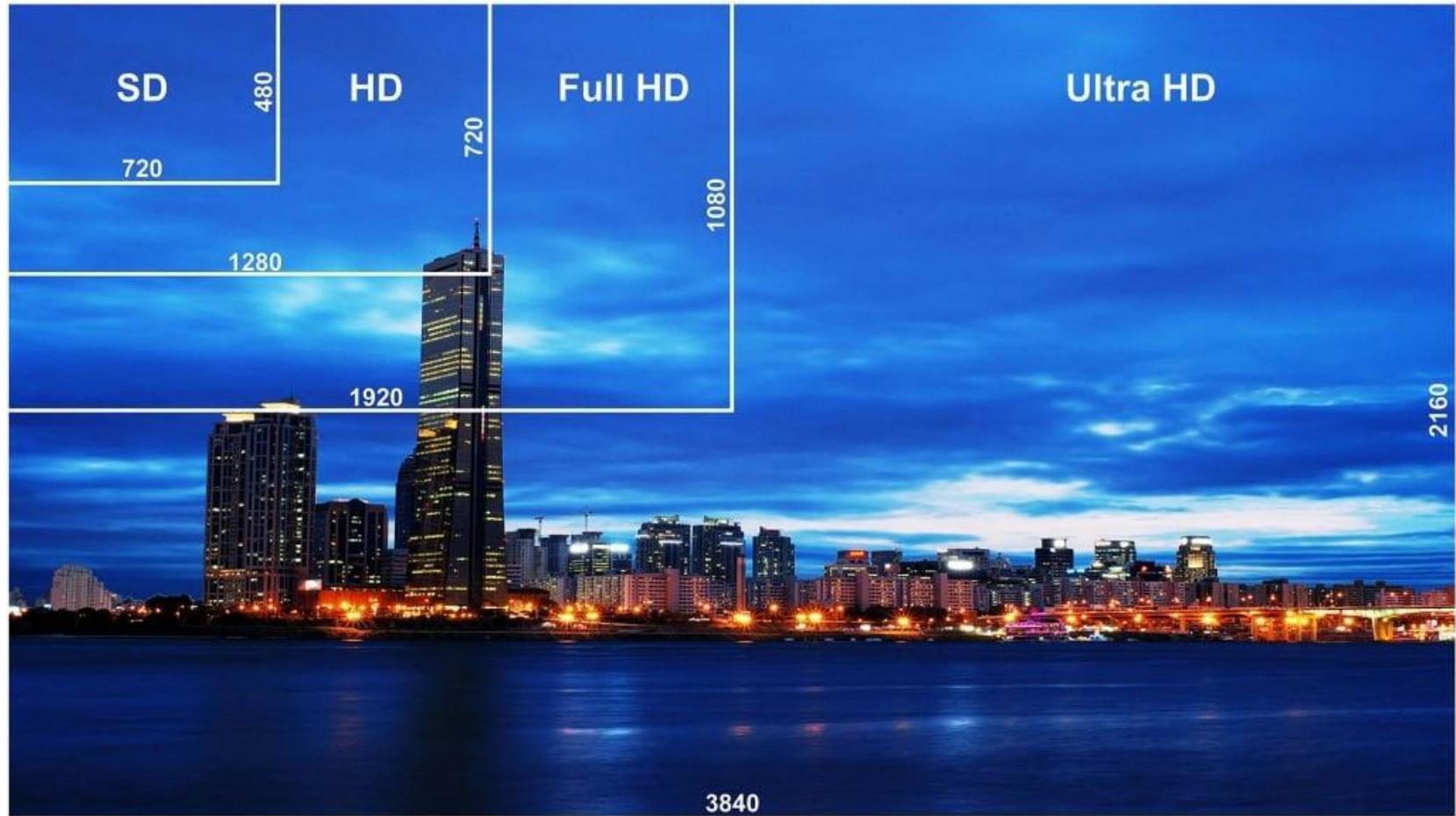
## Misure 4:3

- 640x480 (VGA) a 256 colori ~2.4 Mbit = ~300KB
- 800x600 (SVGA) a 65536 colori ~7.68 Mbit = ~960KB
- 1024x768 (XGA) a 16M colori ~18.87 Mbit = ~2.36MB

## Misure 16:9

- 1280x720 (HD 720) a 16M colori ~ 22.12Mbit = 2.76MB
- 1920x1080 (HD 1080) a 16M colori ~49.77 Mbit = ~6.22MB  
Full HD
- 3840x2160 Ultra HD
- 4096x2160 4k (cinema digitale)

# Diverse risoluzioni e spazio necessario



# Schermi

---

Risoluzione: gli schermi ad alta risoluzione dispongono di una matrice 1920 x 1080: 2 Milioni di pixel. La *bitmap* per rappresentare una immagine a questa risoluzione con 24 bit per indicare il colore (1byte per ciascun colore) richiede uno spazio di 6 Milioni di byte.

RAM dello schermo: l'immagine viene «rinfrescata» da 60 a 100 volte al secondo, per dare l'impressione di una immagine costante. Sul controller dello schermo è presente una memoria dedicata che può contenere più *bitmap* per un passaggio rapido da una immagine ad un'altra.

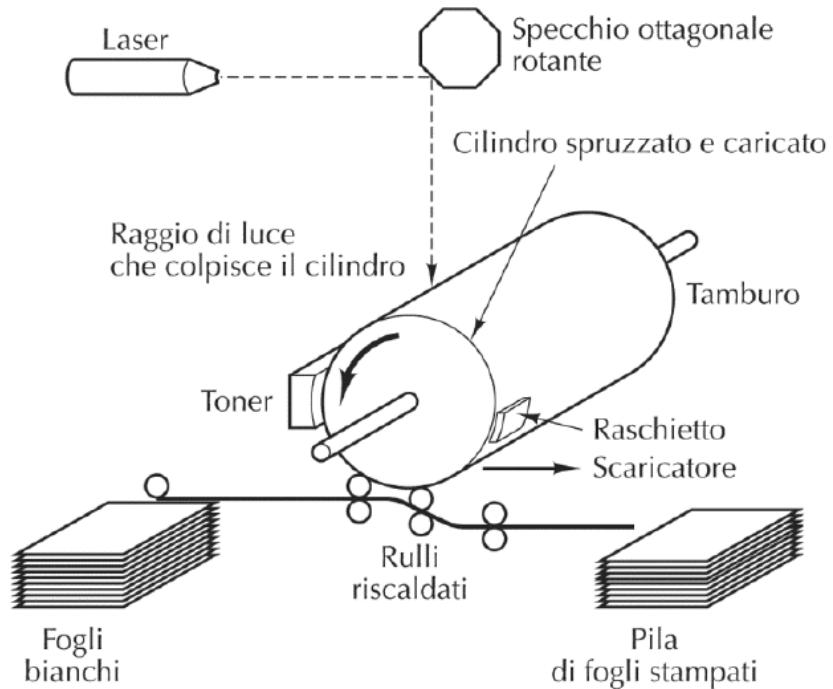
# Schermi: riproduzione video

---

Per trasmettere immagini in movimento, occorre trasferire 25 fotogrammi al secondo, ciascuno da 6Mbyte. E' dunque necessaria una velocità di trasferimento dati di 155Mbyte/sec.

# Stampanti laser (Bianco-Nero)

Risoluzione  
600 dpi  
o  
1200 dpi  
(dpi = dot per inch)

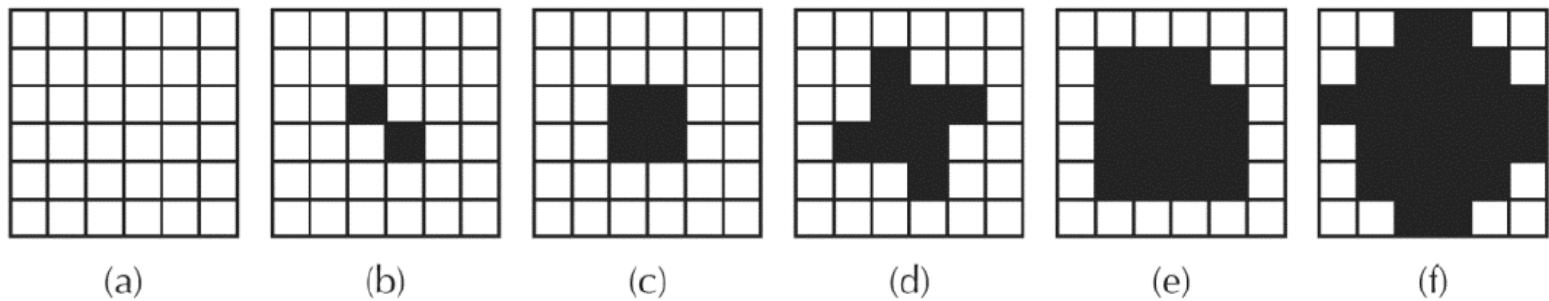


**Figura 2.36** Funzionamento di una stampante laser.

L'immagine trasmessa può essere una bitmap, oppure un file contenente istruzioni per comporre l'immagine (PostScript, PCL, PDF)

# Toni di grigio

---

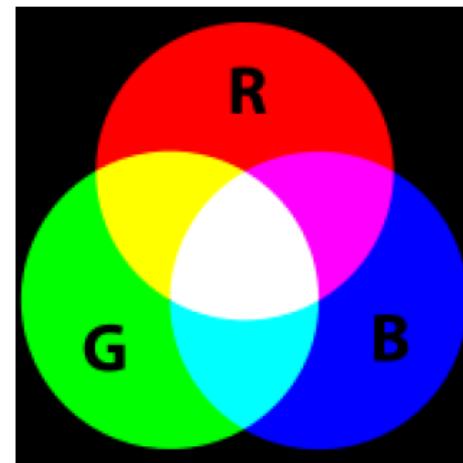
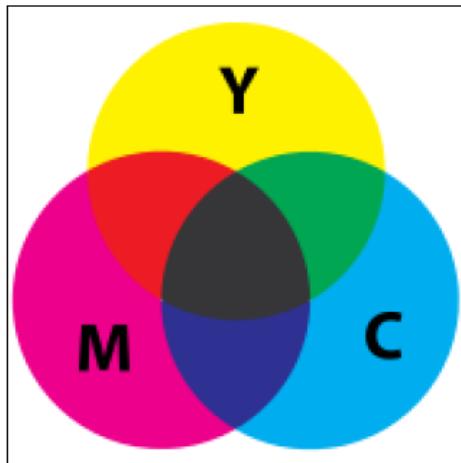


**Figura 2.37** Mezzitoni per alcuni intervalli di livelli di grigio. (a) 0-6. (b) 14-20. (c) 28-34. (d) 56-62. (e) 105-111. (f) 161-167.

Il grigio si ottiene scurendo una certa percentuale di pixel in una matrice 6x6 quindi la risoluzione scende a 100 dpi

# Come si ottengono i colori: stampanti vs schermi

---



Miscelazione sottrattiva dei colori Ciano, Magenta, Giallo (per immagini a luce riflessa)

Miscelazione addittiva dei colori Rosso, Verde, Blu (per immagini a luce trasmessa)

# Stampanti a getto d'inchiostro

---

Hanno una cartuccia per ciascun colore primario sottrattivo e una per il nero. (Si tratta di una stampa in quadricromia CYMK). Risoluzione da 1200 dpi (economiche) a 4800 dpi (di fascia alta).

La testina si muove sulla carta mentre il colore viene spruzzato in piccolissime goccioline dalle cartucce (di dimensioni controllabili tramite un regolatore di tensione)