Input/Output (I/O): il bus, interfacce e periferiche

A. Borghese, F. Pedersini

Dipartimento di Informatica
Università degli Studi di Milano

- Input / Output (I/O): insieme di architetture e dispositivi per il trasferimento di informazione da (OUT) e verso (IN) l'elaboratore
- Dispositivi eterogenei per:
  - velocità di trasferimento
  - latenze
  - sincronismo
  - modalità di interazione (con l'uomo o con la macchina)

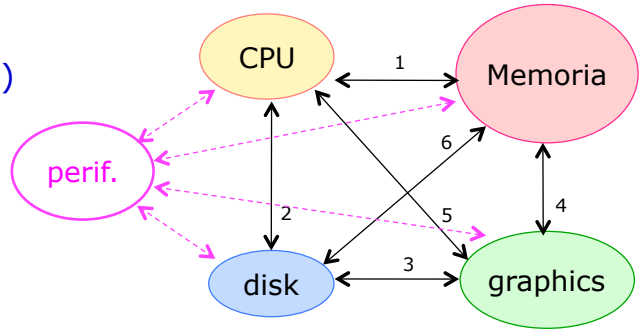
Dispositivo	Comportamento	Interfaccia con...	Data rate [kB/s]
Tastiera / mouse	Input	Umano	0,01 ÷ 0,02
stampante	Output	Umano	100 ÷ 10.000
Floppy disk	I/O (memoria)	Macchina	50
Disco ottico	I/O (memoria)	Macchina	500
Disco magnetico	I/O (memoria)	Macchina	100.000
Rete LAN	I/O	Macchina	10.000 ÷ 1.000.000
Scheda grafica	Output	Umano	> 1.000.000

Collegamento: il *bus*

Come collegare CPU e periferiche?

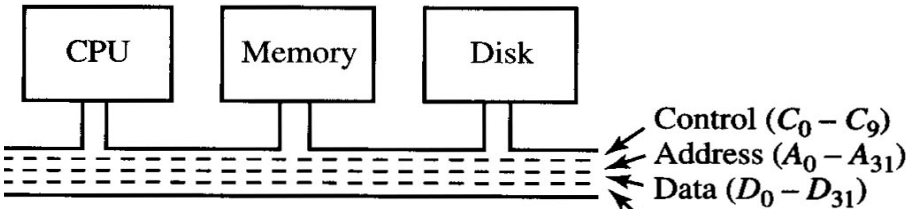
Connessione **completa**? (tutti con tutti)

- N/2 *(N–1) connessioni bidirezionali: complesso e difficile da controllare
- Per ogni nuovo dispositivo: N+1 nuove connessioni

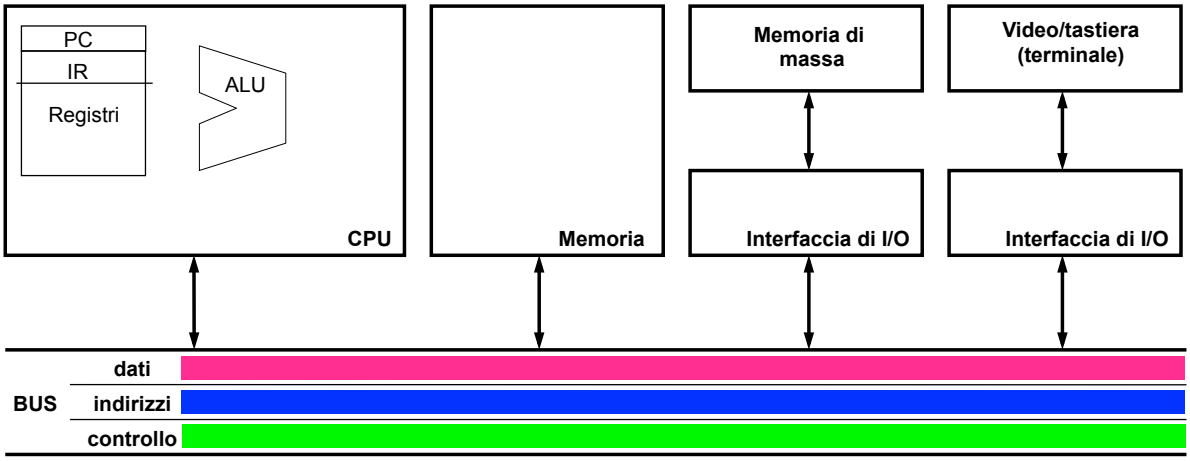


BUS: pathway comune che connette i diversi dispositivi

- Vantaggi: elevata flessibilità, semplicità, basso costo
- Svantaggi: gestione complessa del canale condiviso



Collegamento tra componenti mediante BUS



Connessione *a nodo comune:* ***BUS***

Tutte le unità del calcolatore sono connesse al bus

Bus di sistema

***BUS**: infrastruttura di comunicazione tra diverse unità dell'elaboratore.*

Suddivisione logica in tre sezioni:

- ***Bus dati***: le linee per trasferire dati e istruzioni da/verso i dispositivi.
- ***Bus indirizzi***: su cui la *CPU* trasmette gli indirizzi di memoria (o di periferica) che identificano il dato, in lettura/scrittura dalla memoria (o ingresso/uscita da periferica).
- ***Bus di controllo***: trasporta informazioni ausiliarie per la definizione delle operazioni da compiere e per la sincronizzazione tra i dispositivi.

Esempio: lettura dalla memoria (MemRead)

1. *La CPU fornisce l'indirizzo della parola desiderata sul **bus indirizzi***
2. *viene richiesta l'operazione di **lettura** sul **bus di controllo** (**MemRead**)*
3. *Quando la memoria ha reso disponibile la parola richiesta, il dato viene trasferito sul bus dati e la CPU può prelevarlo dal **bus dati** ed utilizzarlo nelle sue elaborazioni*

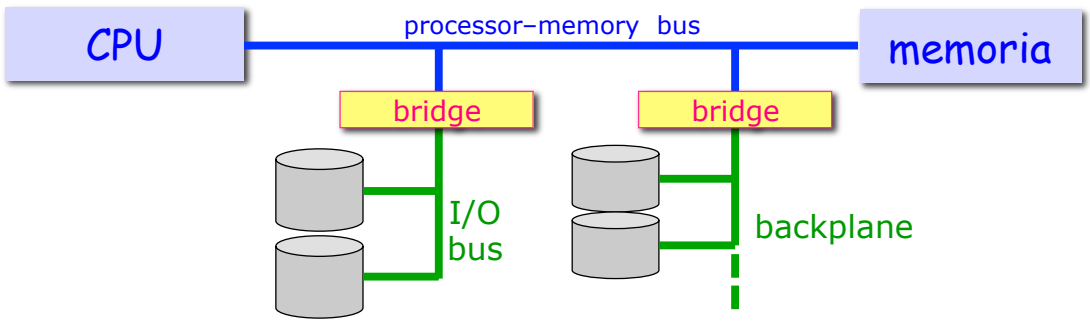
Tipologie di bus

Tipologie di bus nell'elaboratore moderno:

- **Processor–Memory**: lunghezza ridotta, molto veloci (**northbridge / southbridge**)
- **Backplane**: servono per far coesistere la memoria, il processore e i dispositivi di I/O su di un unico bus (ISA, PCI, VME, PC/104)
- **I/O**: notevole lunghezza, molti dispositivi connessi (IEEE-1394, USB, Ethernet)

BRIDGE: dispositivo di adattamento, che permette il collegamento e la comunicazione fra bus differenti.

- Bridge fra system bus (800 MB/s) e PCI bus (133/266 MB/s)
- Bridge fra PCI bus e ISA bus (16.7 MB/s)



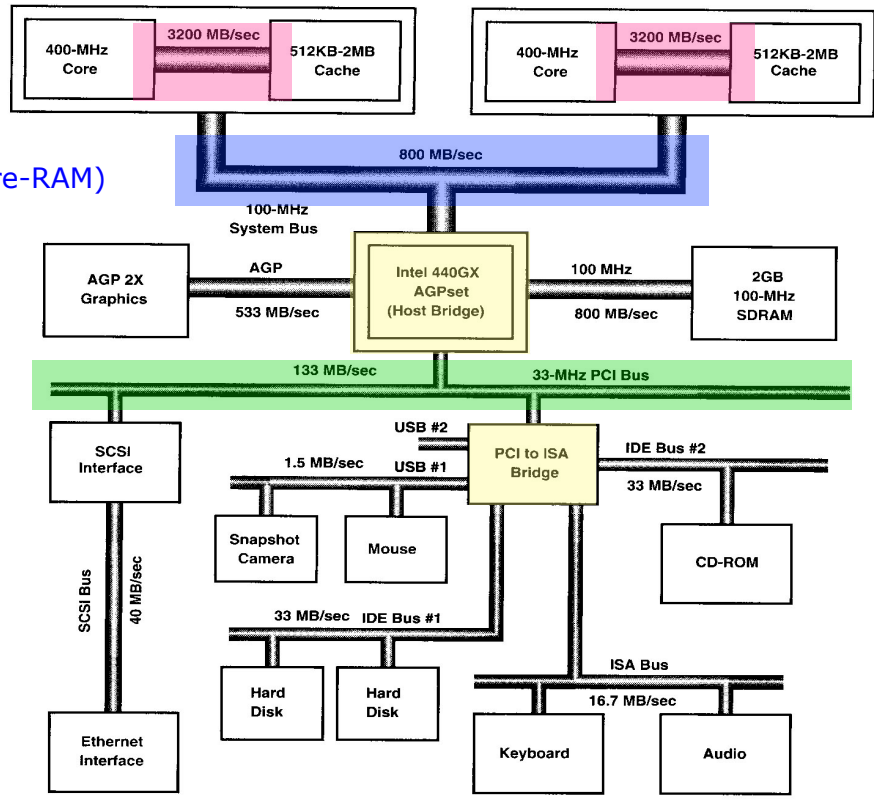
Esempio: **Pentium II – architettura PCI**

Bus **processore-memoria** cache – (3200 MB/s)

Bus di **sistema** (processore-RAM) (800 MB/s)

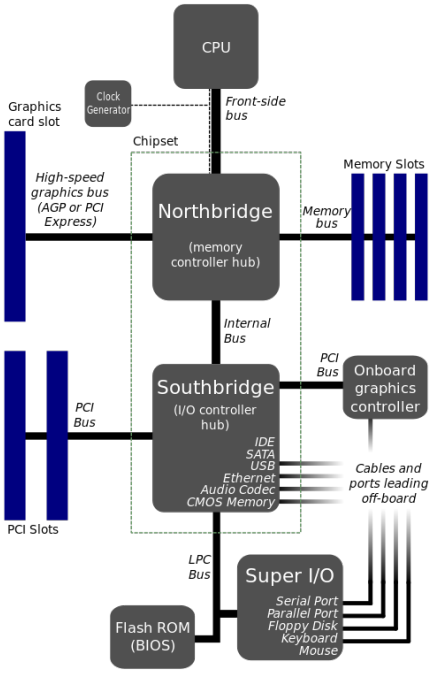
Bus di **backplane**: PCI (133 MB/s)

Architettura basata su bus PCI (da anni '90)

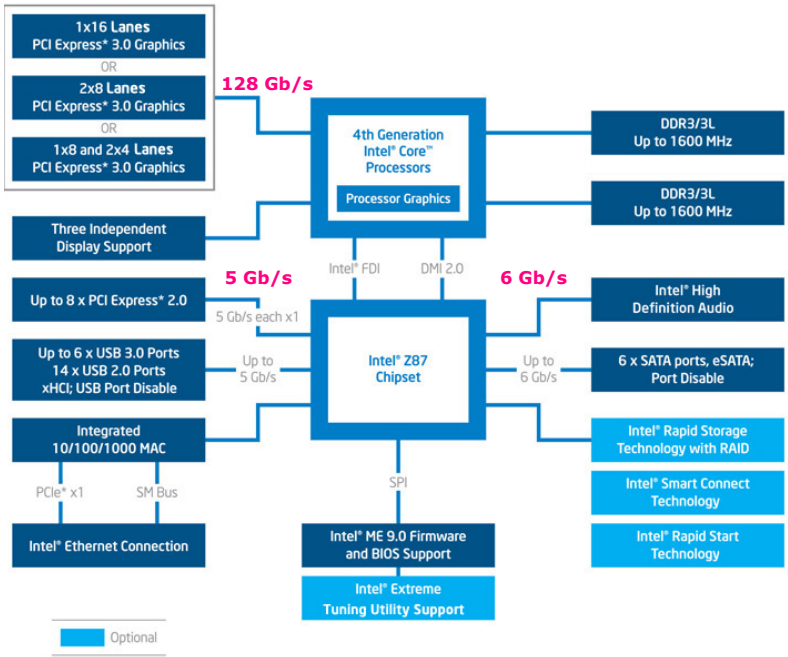


Esempio: **architetture Intel moderne**

Intel chipset architecture (2000 - 2005)



Intel chipset architecture (2013–)



Problematiche di comunicazione su bus

***Problematiche da risolvere** nella comunicazione tramite **bus**:*

1. Sincronizzazione:

Il bus è condiviso, quindi si può "trasmettere" sul bus soltanto uno alla volta
Necessario un meccanismo di sincronizzazione per coordinare la comunicazione

Comunicazione sincrona → **bus sincrono**

- Comunicazione scandita da un segnale di **clock comune**

Comunicazione asincrona → **bus asincrono**

- Non esiste un clock. Sincronizzazione mediante **dialogo** fra i partecipanti. Di norma esiste un **master** che dirige la comunicazione, e uno o più **slave** che obbediscono

2. Controllo di flusso:

I dispositivi collegati al bus hanno **differenti velocità e tempistiche** di risposta e di comunicazione ➔ solitamente all'interno dei dispositivi sono presenti **buffer** ("*tampone*", un magazzino temporaneo) per memorizzare temporaneamente l'informazione e non essere limitati dal dispositivo più lento

Dimensionamento del buffer:

Buffer **grossi** → Aumenta la **velocità di trasferimento**

Buffer **piccoli** → Riduce i **tempi di risposta (latenza)**

Bus sincroni

BUS sincroni:

Il bus è dotato di un segnale di sincronizzazione: **Bus Clock**

Comunicazione scandita dai cicli di Bus Clock

- in generale diverso da quello della CPU, ma sincronizzato con esso

- Esempio: i bus **processore-memoria** sono in genere **sincroni**
  - hanno dimensioni ridotte, pochi elementi connessi e devono essere veloci
  - Ciclo di bus (**bus cycle**): n. cicli per effettuare una transazione: (typ. 2÷5 CC)

Vantaggi

- I protocolli sincroni permettono di ottenere bus molto **veloci**

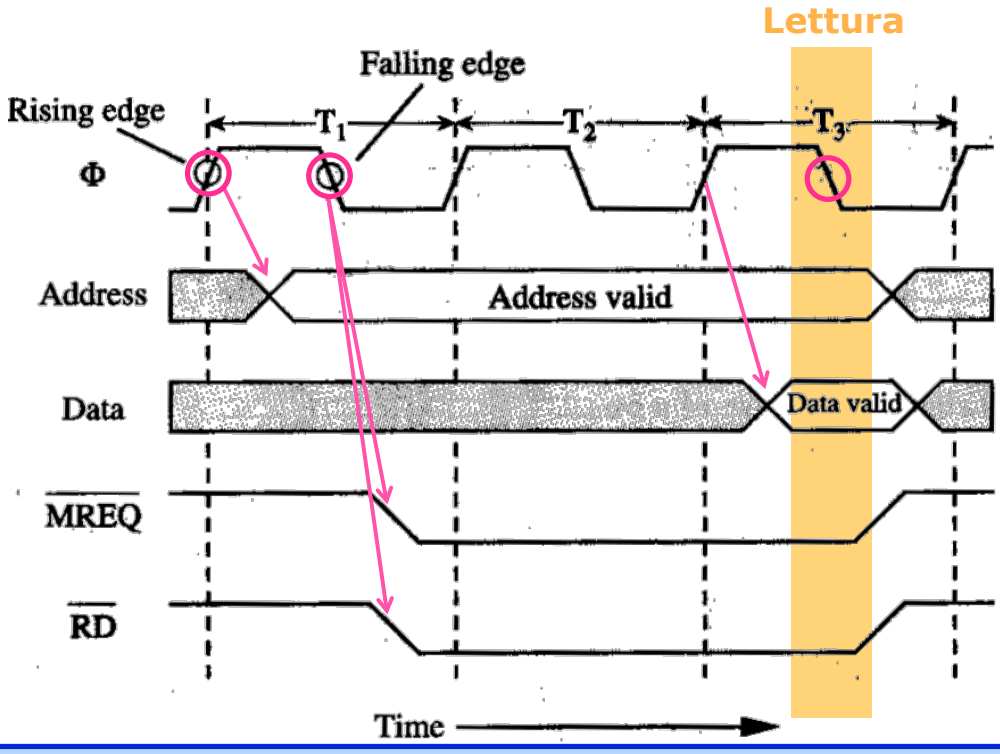
Svantaggi

- Ogni device deve essere sincronizzato → **complessità, alto costo**
- Tutti i dispositivi devono lavorare alla frequenza imposta dal **bus clock**
- **Lunghezza massima limitata** (i ritardi nei fronti dovuti alla propagazione producono disallineamenti – il **bus clock** non è più "istantaneo")

Esempio bus sincrono: Memoria → CPU (MemRead)

Ciclo di **lettura memoria:**

Durata: 3 CC



Esempio bus sincrono: **bus PCI**

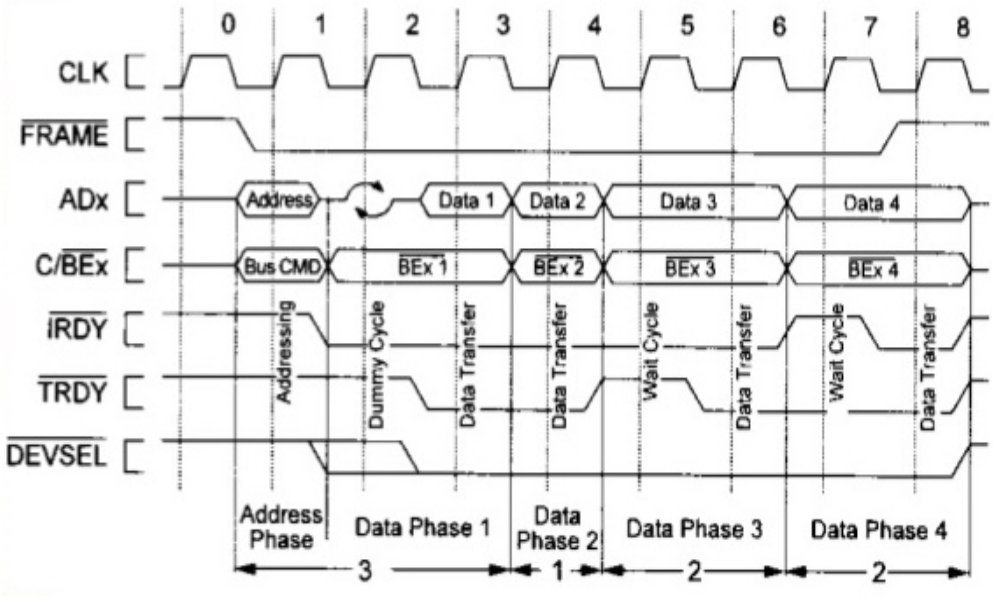
ciclo di lettura

AD: (32 bit) address bus

C/BE: (8÷64 bit) command/data

IRDY: Initiator ready

TRDY: Terminator ready



Bus asincroni

BUS asincroni:

Un bus asincrono non è dotato di clock

La comunicazione avviene mediante un protocollo di **handshaking**

- **Vantaggi**: possono avere lunghezza elevata e connettere molti dispositivi (spesso i **bus esterni** sono **asincroni**)
- **Svantaggi**: in genere, bassa velocità di trasferimento dati

Esempio: comunicazione **Master/Slave**: Master controlla la comunicazione

Segnali: **`Master_Sync:`** **`0:`** `master` **`ready,`** **`1:`** `master` **`release`**
`Slave_Sync:` **`0:`** `slave` **`ready,`** **`1:`** `slave` **`release`**

Protocollo di Handshaking:

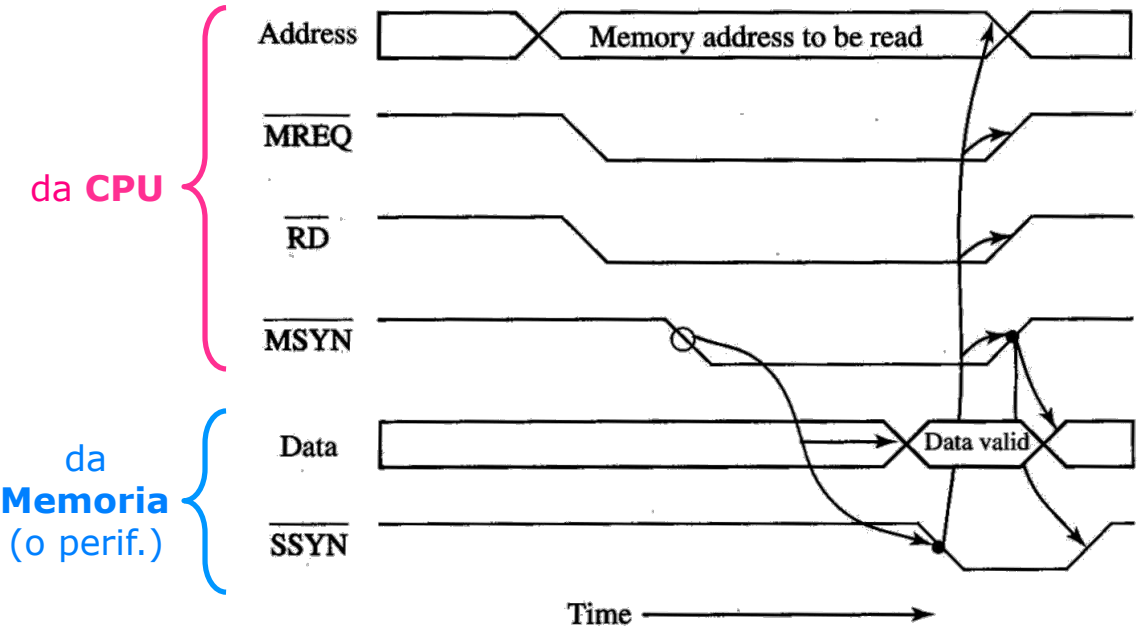
(`MSync`→0) → (`SSync`→0) → **comunic..** → (`MSync`→1) → (`SSync`→1)

Master ready **Slave ready** **Master release** **Slave release**

Bus asincroni

Esempio: lettura memoria, con comunicazione asincrona

(mediante i segnali: **MSYN e SSYN**)



Gestione del bus: *arbitraggio*

*E quando si hanno **più di 2 dispositivi** connessi al bus?*

L'architettura più semplice: **Master/Slave** con **master unico**

- un **unico bus master** ("padrone" del BUS)
- **tutti gli altri** dispositivi sono **slave**

  Tutte le comunicazioni vengono dirette dal master

  - Questo può creare un collo di bottiglia, ad esempio nel caso di trasferimento di dati da I/O a memoria (I/O – CPU ➔ CPU – memoria).

esistono anche...
architetture con più dispositivi master:

- è necessario in questo caso definire (e rispettare!) una policy che coordini i diversi bus master

In questi casi ho bisogno di un criterio di gestione della comunicazione:

➔ **Arbitraggio del bus**

- Questo è il principale inconveniente dei bus a nodo comune.

Protocolli di comunicazione: ***arbitraggio***

Arbitraggio:

Occorre stabilire quale dispositivo autorizzare all'utilizzo del bus

- Arbitraggio **centralizzato**:
  arbitro unico, che decide per gli altri
- Arbitraggio **distribuito**:
  decisione distribuita nel sistema

Meccanismo di accesso al bus:

1. Richiesta del bus: **BUS REQUEST**
2. Assegnamento del bus: **BUS GRANT**

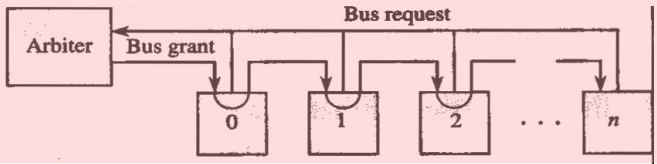
Problema: assicurare sia *fairness* che *efficienza*

- Qualità contrastanti – si cerca un **compromesso**

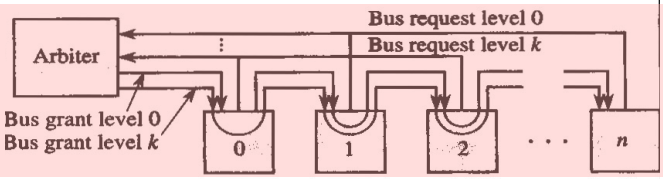
Schemi di arbitraggio

Schemi centralizzati

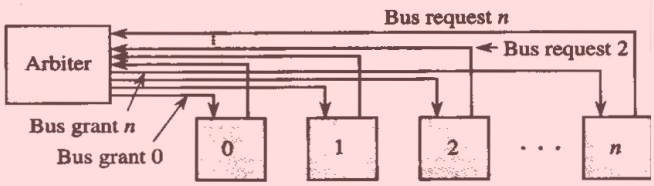
Arbitraggio Daisy Chain



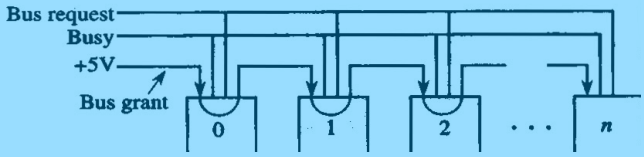
Arbitraggio Centralizzato con priorità



Arbitraggio Centralizzato Parallelo



Schema Decentralizzato (distribuito)



Alcuni tipi di bus...

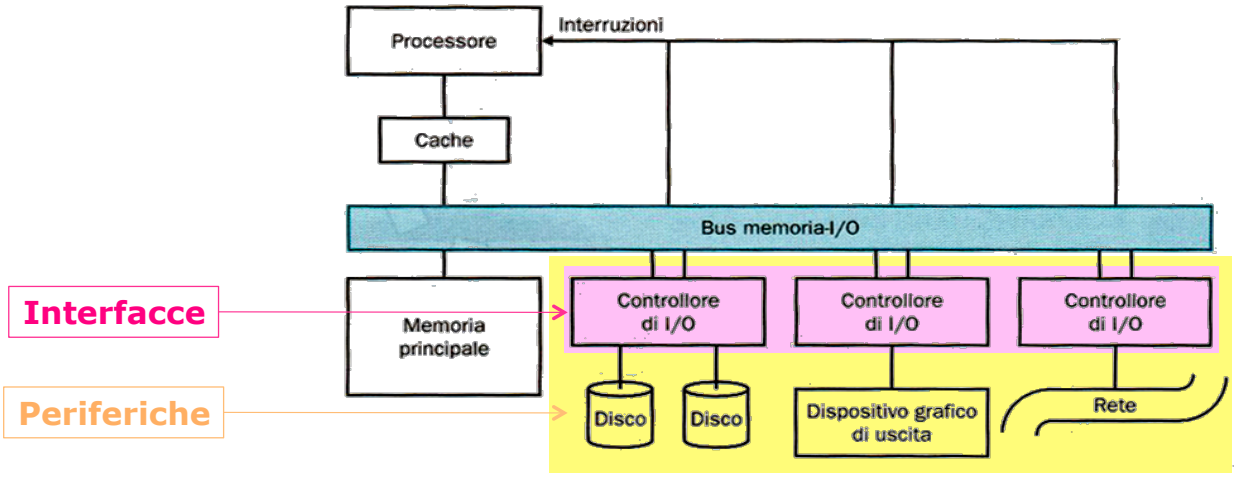
	PCI	Firewire	SCSI
Nome dello standard	**PCI**	**IEEE 1394**	**ANSI X3.131**
Tipo di bus	Backplane	Backplane & I/O (peer-to-peer)	I/O
Ampiezza bus (n. segnali del bus dati)	32-64	Seriale	8-32
Numero di dispositivi master	molti	molti	molti
Temporizzazione	Sincrono 33-66Mhz	Asincrono o sincrono	Asincrono o sincrono (5-20Mhz)
Banda di picco teorica	133-512MB/s (PCI64)	400-800MB/s (IEEE 1394)	5-160 MB/s
Banda stimata raggiungibile	80 MB/s	266-533 MB/s	2,5-160 MB/s
Max numero di dispositivi	1024 (32 disp./segm.)	63	7-31
Max lunghezza del bus	0,5 metri	4,5 metri	25 metri

Le periferiche

I/O: le periferiche

PERIFERICA: Dispositivo per ingresso/uscita di informazioni, collegato alla CPU mediante **bus** e **interfacce**

INTERFACCIA:
Procedura standardizzata (hw/sw: circuiti/protocolli) per la comunicazione
- **Hardware**: circuito "**controller**" della periferica
- **Software**: programma "**driver**" della periferica



Interfacce: tassonomia

Classificazione delle interfacce

Modalità di trasferimento dati:

- **Seriale**: **1 bit** alla volta (RS-232, Ethernet, USB, Firewire)
- **Parallela**: **più bit** alla volta (Centronics, SCSI, IDE)

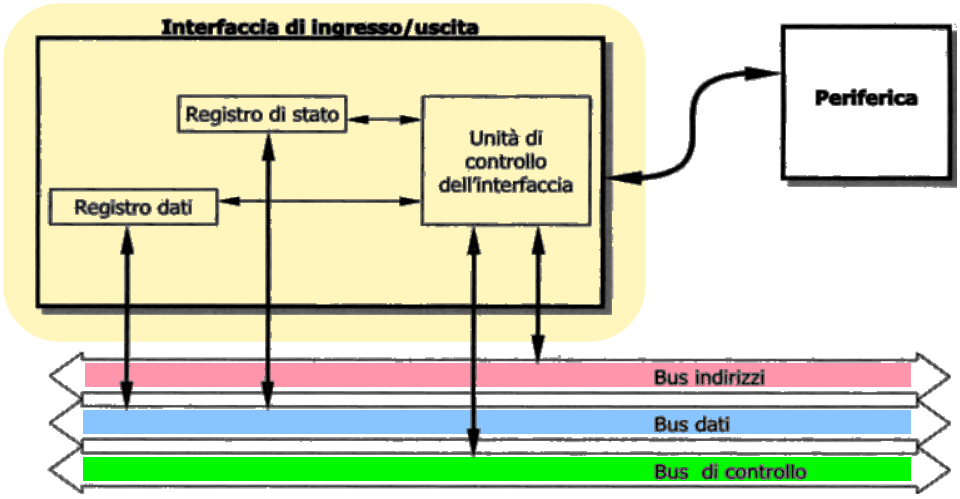
Topologia del collegamento

- **Punto-punto** (RS-232, Firewire, …)
- **A canale condiviso (bus)** (Ethernet, Bluetooth, USB, …)

Mezzo trasmissivo:

- **Via cavo** Cavi in rame, Fibre ottiche
- **Wireless** Infrarossi (IrDA),
  Onde radio (Bluetooth, WiFi, ZigBee, NFC)

Struttura di interfaccia di una periferica



Struttura di un'interfaccia

sostanzialmente, è una macchina a stati finiti:

- **Registri Dati (buffers)**
- **Registri di Stato**

  situazione (stato) della periferica (idle, busy, down….), fase del comando in esecuzione.

Indirizzamento dell'I/O

Come si accede alle periferiche, da programma?

- Due modalità fondamentali:

1 Memory-mapped

- Come se si accedesse a celle della memoria principale
- Necessità di distinguere tra indirizzi di memoria ed indirizzi di I/O

2 Mediante **istruzioni speciali di I/O**

- Istruzioni-macchina (fanno parte della ISA della CPU) che permettono di accedere direttamente a un dispositivo di I/O
- Nell'istruzione è contenuto:
  - Numero identificativo del dispositivo (indirizzo di I/O)
  - Parola di comando (o indirizzo della parola di comando)

Indirizzamento *Memory-mapped*

Memory mapped: i registri del device controller sono considerati come celle di memoria RAM.

- I loro indirizzi saranno diversi da quelli delle celle di memoria.

- Il processore esegue operazioni di I/O come se fossero operazioni di lettura/scrittura in memoria.
  - *Esempio:* `sw $s0, indirizzo`
    `lw $s0, indirizzo`
  - dove **`indirizzo`** è al di fuori dallo spazio fisico della memoria

Istruzioni di I/O: attivano sul bus segnali r/w per le periferiche e non per la memoria ➔ non c'è bisogno di differenziare gli indirizzi

Esempi:
```
lw/sw $s0, 0x80   # r/w memoria, ind:0x80
out $s0, 0x80     # r/w periferica, ind:0x80
```


Indirizzamento Memory-mapped

I/O MIPS32: Memory mapped

- I controller delle periferiche I/O ascoltano tutti i segnali in transito sul bus (*bus snooping*)…
- …e si attivano solamente quando riconoscono sul bus indirizzi l'indirizzo corrispondente alla propria locazione di memoria.

Gestione indirizzi di I/O in modalità *memory mapped*

- Gli indirizzi riservati ai registri del controller corrispondono di norma a porzioni di memoria riservate al S.O.
  ➔ **non sono accessibili quindi al programma utente.**
- I programmi utente devono quindi passare dal S.O. per accedere a questi indirizzi riservati (modalità: **kernel**)
- Questo è quanto viene fatto ricorrendo alle **`system call`**

Tecniche di controllo dell'I/O

- Tecniche di controllo I/O
  - a controllo di programma diretto
  - a controllo di programma con polling
  - ad interruzione (interrupt)
  - ad accesso diretto alla memoria (DMA)
- Memorie di massa: tecnologie

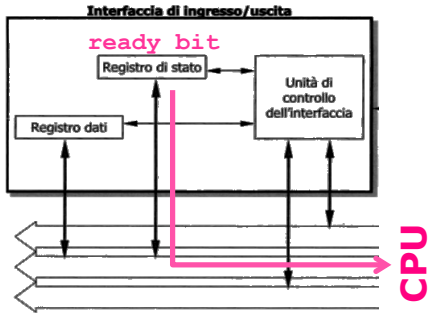
I/O a **controllo di programma**

I/O a **controllo di programma:**

- **La periferica** ha un ruolo **passivo** ➔ fa tutto la CPU
- **La CPU** si occupa sia del **controllo**, sia del **trasferimento dati**.
  - **La CPU**, dopo avere comunicato al controller l'esecuzione dell'I/O, **si ferma** e si mette ad **interrogare il registro di stato** della periferica **in attesa** che il `ready bit` assuma un determinato valore.

Vantaggio: risposta veloce al `ready bit`
Svantaggio: CPU bloccata in stato di: ***busy waiting*** o ***spin lock***

```
begin
   ... // Predisponi i registri del controller
   ... // ad effettuare una operaz. di lettura
   while (ready-bit == 0) do { }; // spin lock
   ... // Carica il dato acquisito
end;
```



I/O a controllo di programma – costo

*Esempio: **tastiera** gestita a controllo di programma*

- Tastiera gestita a controllo di programma che genera **10 Byte/s**
- Frequenza di clock CPU: **50 MHz**

Determinare il tempo in cui verrebbe effettivamente utilizzata la CPU per trasferire **1 parola**, tenendo conto che ci vogliono **20 cicli di clock per ogni byte**

T – tempo di trasferimento di una parola:

T = 4 byte/parola **/ 10** Byte/sec **= 0,4** sec/parola

#cicli_clock trascorsi / parola = $50 \cdot 10^6 * 0,4 = \mathbf{2 \cdot 10^7}$ cicli_clock/parola

In realtà si utilizzano per la tastiera solo:

$20 \cdot 4 = \mathbf{80}$ cicli_clock per trasferire la parola

→ **sfruttamento CPU:** $(80 / 2 \cdot 10^7) = 4 \cdot 10^{-6} = 0,0004$ %

- **Prestazioni I/O a controllo di programma:**
  - \+ Risposta molto veloce (**bassissima latenza**)
  - – Alta probabilità di **spin lock**

Sommario

- Tecniche di controllo I/O
  - a controllo di programma diretto
  - a controllo di programma con **polling**
  - ad interruzione (interrupt)
  - ad accesso diretto alla memoria (DMA)
- Memorie di massa: tecnologie

Polling

POLLING: interrogazione **ciclica** (non bloccante) dello stato di tutte le periferiche

- Interrogazione del **registro di stato** della periferica
- In presenza di uno stato di **busy–waiting** su una periferica, si esegue il **polling sulle altre** periferiche di I/O
- Se la periferica interrogata necessita di intervento:
  - si soddisfa la richiesta...
  - ...quindi si prosegue il ciclo di polling sulle altre periferiche

```
// Leggi dato da perif_x
begin
    Predisponi i registri controller per una read;
b:  if (ready_bit(perif_1) == 1) servi perif_1;
    if (ready_bit(perif_2) == 1) servi perif_2;
        …
    if (ready_bit (perif_n) == 1) servi perif_n;
    goto b;
end;
```


I/O con polling: valutazione prestazioni

Polling dura di più del controllo da programma, ma non è bloccante!

1. trasferimento del controllo alla procedura di polling,
2. accesso alla periferica,
3. ritorno al programma utente

Tipico: ~ **400 cicli clock**

Prestazioni I/O con polling:

+ Risolve il problema dello spin lock
– Risposta meno veloce che nel controllo di programma (**maggiore latenza**)

Valutazione efficienza: *data una CPU con clock f_C=500 MHz, determinare l'impatto del polling per 3 dispositivi diversi:*

- **Mouse**: deve essere interrogato almeno 30 volte al secondo per non perdere alcun movimento dell'utente.
- **Floppy disk**: trasferisce dati al processore in parole da 16 bit ad una velocità di 50 kByte/s
- **Hard disk**: trasferisce dati al processore in blocchi di 4 parole alla velocità di 4 MByte/s

Efficienza I/O mediante Polling

Efficienza: % utilizzo della CPU per gestire il trasferimento I/O

Grazie al polling non bloccante, nel tempo restante **la CPU può fare altro**

- Mouse:
  - 30 x 400 = 12,000 cicli_clock/sec
  - 12,000 / 500,000,000 = 0,000024 sec/sec → **0,0024%**
  - *Piccolo impatto sulle prestazioni*
- Floppy disk:
  - Per ogni accesso possiamo trasferire **2 byte**
  - A 50 kB/sec → occorrono **25 k accessi/sec**
  - In termini di cicli di clock: 25k x 400 = **10 Mcicli_clock/sec**
  - (ignorando la differenza tra 25k e 25.000) 10 M / 500 M → **2%**
  - *Medio impatto sulle prestazioni*
- Hard disk:
  - Per ogni accesso possiamo trasferire **16 Byte**
  - a 4 MB/sec → occorrono **250 kaccessi/sec**
  - In termini di cicli di clock: 250k x 400 = **100 Mcicli_clock/s**
  - (trascurando la differenza tra 1k e 1000) 100M/500M → **20%**
  - *Alto impatto sulle prestazioni*

Polling e Controllo diretto: limiti

Limiti di: polling e controllo di programma diretto

- **Controllo diretto**: alta probabilità di **spin lock**
- **Polling**: durata talvolta notevole della procedura di gestione (**overhead**)

In entrambi i casi...

- con periferiche lente:
  - eccessivo spreco del tempo di CPU, che per la maggior parte del tempo rimane occupata nel ciclo di **busy waiting / polling**
- con periferiche veloci:
  - il lavoro svolto dalla CPU è quasi interamente dovuto all'effettivo trasferimento dei dati

Sommario

- Tecniche di controllo I/O
  - a controllo di programma diretto
  - a controllo di programma con polling
  - ad interruzione (interrupt)
  - ad accesso diretto alla memoria (DMA)
- Memorie di massa: tecnologie

Interrupt

`interrupt`:
interruzione del normale funzionamento del processore, da parte di un'entità esterna e indipendente (periferica)

La **periferica segnala alla CPU** di avere bisogno di attenzione mediante un apposito segnale (nel bus di controllo)

Funzionamento del meccanismo di interrupt:

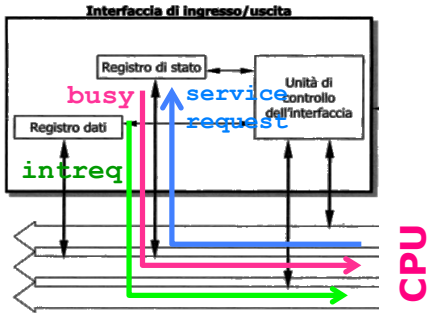
1. La periferica che necessita di attenzione avvisa il processore, attivando il segnale: **`interrupt request`**
2. Quando il processore "se ne accorge" (fase di fetch), informa di averlo ricevuto mediante il segnale: **`interrupt acknowledge`**
3. Il processore abbandona l'esecuzione del programma corrente e passa ad eseguire la **procedura di risposta all'interrupt**
4. Terminata la procedura di risposta, il processore riprende l'esecuzione del programma interrotto (istruzione: return from interrupt – **`RetI`**)

Il programma utente deve in seguito poter procedere dal punto in cui è stato interrotto ➔ è necessario effettuare un ***salvataggio del contesto***

Esempio: comando **print** con periferica **busy**

Esempio: gestione di una stampa mediante **`interrupt`**

1. Invio del comando: **`print`** alla periferica di stampa
2. Se la periferica è in stato **`busy`**, CPU torna alla sua attività, ma prima scrive nel registro di controllo la richiesta di output: **`service request`**
3. Quando la periferica diventa **`ready`**, viene inviato un interrupt → **`interrupt request`**
4. Il programma di risposta all'interrupt provvederà a trasferire alla periferica il dato che si vuole stampare
5. Terminato di "servire" l'interrupt, riprende l'esecuzione del programma originario, dal punto di interruzione



I/O ad interrupt - Valutazione efficienza

Esempio: ***I/O*** *da* ***hard disk*** *con* ***interrupt***

- Frequenza di clock è **500Mhz**
- Trasferimento di blocchi di **16 byte (4 word)** ad ogni interrupt
- Trasferimento a **4 Mbyte/s**
- Il costo di ogni interruzione è **500 cicli di clock**
- Il disco trasferisce dati in media solamente per il **5%** del suo tempo

- Trasferimento:
  4 MB/s = 1Mword/s → 250 k interrupts/sec
- Costo dell'interruzione:
  *250 k int/s * 500 cicli_clock = 125 Mcicli_clock/sec*
- Frazione di utilizzo del processore per il trasferimento (interrupt):
  125 M / 500 M = 25 %
- Frazione del processore utilizzata in realtà (5%):
  *125M / 500M * 0.05 = 1,25 %*

Interrupt **gerarchici**

In un elaboratore devo gestire **molti interrupt di natura diversa**

Posso gestire interrupt multipli in due modi:

- Accodamento degli interrupt → **FIFO**
- Annidamento degli interrupt → **LIFO** (cf. stack)

Occorre definire una priorità!

Maschere di interrupt

- Ad ogni interrupt viene associato un livello
- Il **livello corrente** è associato allo stato del sistema
- Interrupts **allo stesso livello** sono accodati **FIFO**
- La maschera dell'interrupt viene memorizzata nello **status register** (registro di stato del sistema – 8 livelli in MIPS)
- La maschera degli interrupt pending viene caricata nel cause register. Per ogni bit a 1, c'è un interrupt pending per quel livello
- Codifica di priorità

Sommario

- Tecniche di controllo I/O
  - a controllo di programma diretto
  - a controllo di programma con polling
  - ad interruzione (interrupt)
  - ad accesso diretto alla memoria (**DMA**)
- Memorie di massa: tecnologie

DMA: Direct Memory Access

Limiti della gestione interrupt-driven:

La gestione mediante interrupt non svincola il processore dal dover eseguire le operazioni di trasferimento

[PERIFERICA ➔ CPU] → [CPU ➔ MEMORIA]

Per periferiche veloci, le operazioni di trasferimento dati occupano un tempo preponderante rispetto al tempo speso in spin-lock

Per evitare l'intervento della CPU nella fase di trasferimento dati, è stato introdotto il protocollo di trasferimento:
Direct Memory Access (DMA)

Caratteristiche della DMA

DMA controller: processore specializzato nel trasferimento dati tra dispositivi di I/O e memoria centrale

- Per attivare il trasferimento, la CPU trasmette al DMA controller le seguenti informazioni:
  - il tipo di operazione richiesta
  - l'indirizzo di partenza dell'area in cui trasferire i dati
  - il numero di bytes riservati in memoria centrale

- Il **DMA controller gestisce il trasferimento** dei dati: la **CPU si svincola** completamente dall'operazione di I/O

- **Al termine** del trasferimento dati, il DMA controller invia un **`interrupt request`** alla CPU per segnalare il completamento del trasferimento. Nella procedura di risposta all'interrupt, la CPU riprende il controllo del bus.

I/O mediante DMA – efficienza

Esempio: **I/O** da **hard disk** con **DMA** (di blocchi da 8kB)

- Frequenza di clock è 500 Mhz
- Trasferimento di blocchi di **8 kbyte** per ogni **DMA**
- Trasferimento a 4 Mbyte/s
- Il costo dell'inizializzazione del DMA è di **1000** cicli di clock
- Il costo dell' interruzione al termine del DMA è di **500** cicli di clock

Per ciascun trasferimento DMA occorre:

- Setup: 1000 + 500 cicli di clock (tempo di inizio + tempo di fine)
- Transfer DMA (a 4Byte/CC): 8kB/4B = 2000 CC
- Numero di DMA/sec: 4 Mbyte/s / 8 kbyte = 500 DMA/sec
- N. cicli clock CPU richiesti: (1500+2000)*500 = 1,75 MCC/sec
- Frazione del processore utilizzata: 1,75M / 500M = **0,35 %**

Sommario

- Tecniche di controllo I/O
  - a controllo di programma diretto
  - a controllo di programma con polling
  - ad interruzione (interrupt)
  - ad accesso diretto alla memoria (DMA)
- Memorie di massa: tecnologie

Memorie di massa: tecnologie

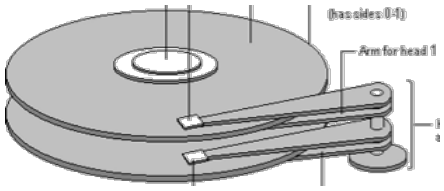
Memorie di massa: memorie persistenti, atte a memorizzare dati in modo non volatile

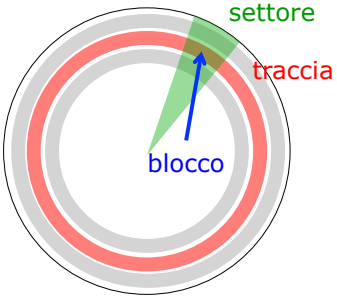
Tecnologie:

- Memorizzazione **MAGNETICA**
  - I dati sono letti/scritti mediante una testina magnetica (che genera un intenso campo magnetico) su un supporto di materiale ferromagnetico (che "memorizza" il campo magnetico cui è stato sottoposto)
  - DISCHI (hard disk, floppy disk), NASTRI
- Memorizzazione **OTTICA**
  - I dati sono letti/scritti mediante una raggio laser su un supporto riflettente (che riflette ("1") o meno ("0") il fascio di luce laser)
  - CD–ROM, DVD, Blu-ray
- Memorizzazione a **SEMICONDUTTORE**
  - FLASH memories

Hard disk – disco rigido

- Costituiti da piatti rotanti (da 1 fino a 25) ognuno con due facce
  - Esiste una testina per ogni faccia
  - Le testine di facce diverse sono collegate tra loro e si muovono in modo solidale tra loro
  - La pila dei piatti viene fatta ruotare a velocità costante: (**4,200 – 10,000 rpm**)
- Ogni faccia è divisa in circonferenze concentriche chiamate tracce
  - Solitamente, ogni traccia contiene la stessa quantità di bit
  - le tracce più esterne memorizzano informazione con densità minore
- L'insieme delle tracce di ugual posto su piatti diversi è chiamato cilindro
- Ogni traccia è suddivisa in settori
  - Il **blocco** è la più piccola unità che può essere letta/scritta su disco (512B ÷ 8kB)





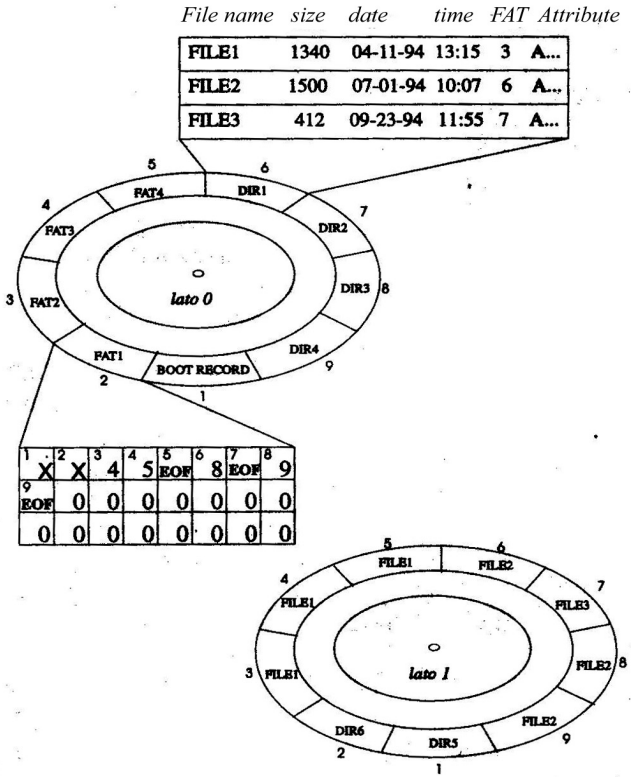
Hard disk – struttura

L'indice del contenuto del disco è solitamente scritto nella **traccia 0**

*File Allocation Table (**FAT**)*

Primo settore:
Master Boot Record (**MBR**)

- dedicato a informazioni strutturali
  - programma di boot
  - definizione delle partizioni
  - tipo di file system



Hard disk – lettura / scrittura

- Per leggere/scrivere informazioni sono necessari tre passi:
  - la testina deve essere posizionata sulla traccia corretta;
  - il settore corretto deve passare sotto la testina;
  - i dati devono essere letti/scritti
- **Tempo di seek (ricerca)**
  - tempo per muovere la testina sulla traccia corretta.
- **Tempo di rotazione**
  - tempo medio per raggiungere il settore da trasferire (tempo per 1/2 rotazione).
- **Tempo di trasferimento**
  - tempo per trasferire l'informazione.
- A questi tempi va aggiunto il tempo per le operazioni del controller.

Sono **tempi medi**, perché volta per volta,
il tempo impiegato è causale.

Hard Disk – prestazioni

IBM/Hitachi DeskStar 7K250 (2004)

- Prestazioni
  - Capacità: 160 Gbyte
  - Buffer (cache) Size: 8 MBytes
  - Spindle Speed 7,200 RPM
  - Transfer Rate (max): 150 MB/sec
  - Average Seek Time, R/W: 8.5/15 ms typ.
  - Track-to-Track Seek, R/W: 1.1 ms typ.
  - Average Latency: 4.17 msec
  - Prob. d'errore (non recuperabile): 10^{-14}
- Caratteristiche fisiche:
  - Number of Discs (physical): 3
  - Number of Heads (physical): 4
  - Total Cylinders: 16,383
  - Dimensioni: 101.6 x 146.1 x 25.4mm
  - Peso: 0.6 kg

IBM/Hitachi UltraStar 7K250 (2016)

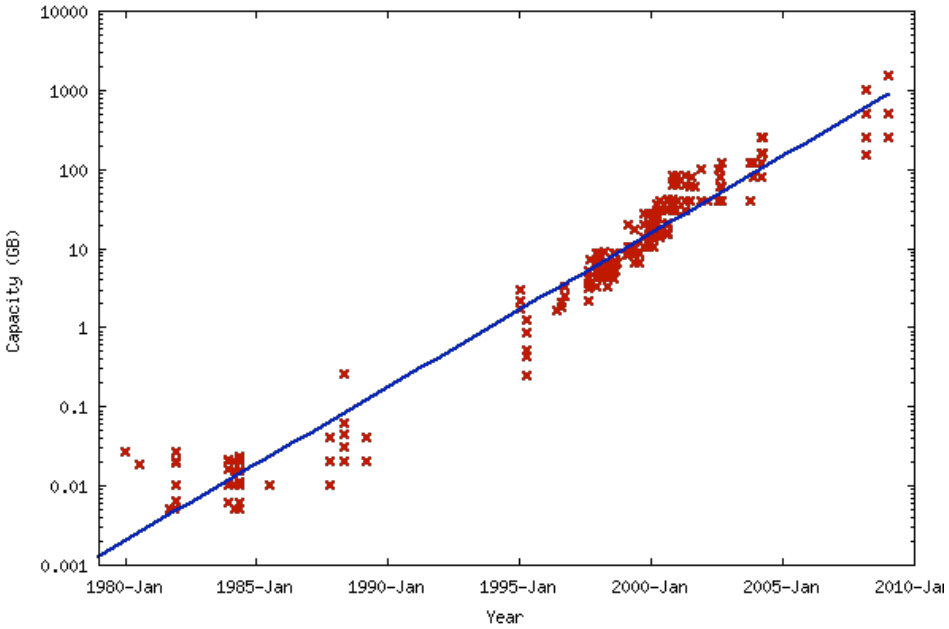
- Prestazioni
  - Capacità: 10 Tbyte **(62x)**
  - Buffer (cache) Size: 256 MBytes **(32x)**
  - Spindle Speed 7,200 RPM
  - Transfer Rate (max) 600 MB/sec **(4x)**
  - Average Seek Time, R/W: 8.0/8,6 ms typ.
  - Track-to-Track Seek, Read-Write: 1.1 ms typ.
  - Average Latency: 4.17 msec
  - Prob. d'errore (non recuperabile): 10^{-15} **(10x)**
- Caratteristiche fisiche:
  - Number of Discs (physical): up to 7 **(2,3x)**
  - Number of Heads (physical): up to 16 **(4x)**
  - Total Cylinders: 16,383

Hard Disk – prestazioni

Aumento di capacità hard-disk magnetici nel tempo

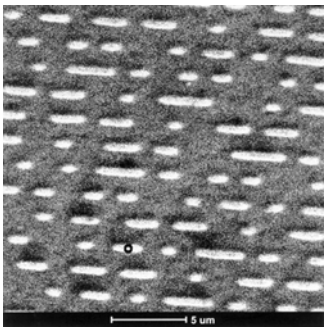
esponenziale: fattore 10 ogni 5 anni

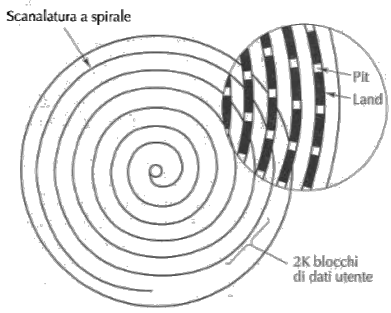
(fonte: Wikipedia)

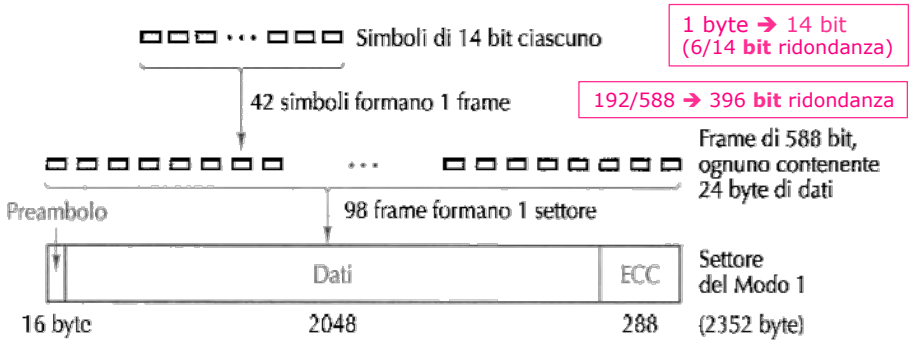


CD–ROM

- Struttura: superficie metallica protetta da uno strato di policarbonato
  - **LETTURA**: Un raggio laser infrarosso (λ=780 nm) colpisce la superficie del disco e viene riflesso in modo diverso a seconda dell'incisione (sì/no ➔ 0/1)
  - Capacità: **700 MByte**
  - Ampiezza pista: **1,6 μm**, lunghezza: **5,6 km**



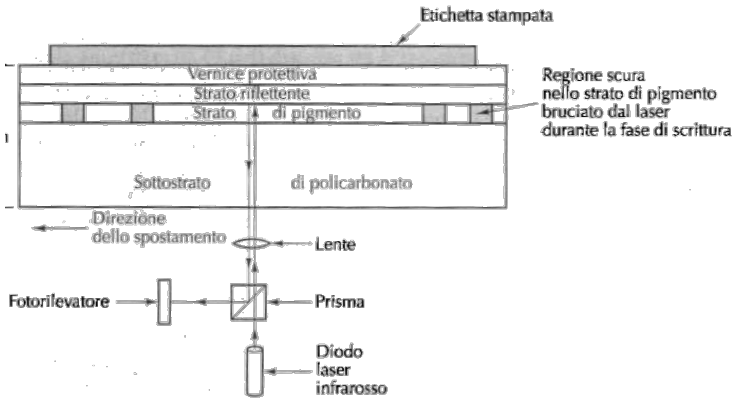




DVD

CD–R (recordable)

- **SCRITTURA**: la superficie metallica, riflettente, è coperta da un pigmento
- Il pigmento diviene opaco quando "bruciato" dal laser.



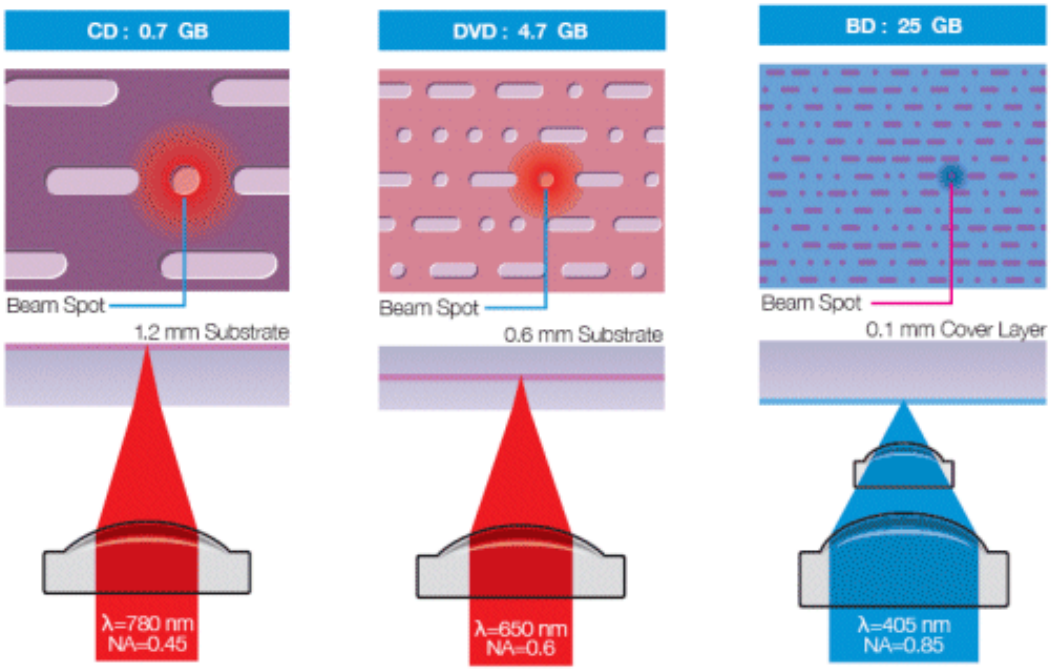
DVD (Digital Video Disk)

- Evoluzione tecnologia, sia per laser che per supporto: lettura/scrittura dati anche in profondità nel metallo (non solo in superficie)

Blu-ray DVD: utilizza un laser blu; capacità fino a **50 GB**

	CD	DVD	Blu-ray
largh. traccia:	**1,6 μm**	**0,75 μm**	**0,32 μm**
λ raggio laser	**IR: 0,78 μm**	**red: 0,65 μm**	**blue: 0,405 μm**
Capacità:	**0,7 GB**	**4,75 ÷ 17 GB**	**25 ÷ 50 GB**

CD / DVD / Blu-ray



Sorgente:

Memorie FLASH

- Cella: 1 transistore MOS con doppio GATE:
  - **Control gate** e **Floating gate** (isolato)
- Operazioni:
  - **SCRITTURA**: iniezione di carica nel floating gate (*"hot injection"*)
  - **CANCELLAZIONE**: svuotamento della carica (effetto tunnel)