Lezione 29

Input/Output: bus, interfacce, periferiche

A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Input/Output

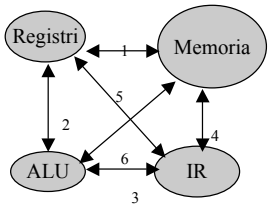
- Input / Output (I/O): insieme di architetture e dispositivi per il trasferimento di informazione da (Out) e verso (In) l'elaboratore
- Dispositivi eterogenei per:
  - velocità di trasferimento
  - latenze
  - sincronismo
  - modalità di interazione (con l'uomo o con la macchina)

Dispositivo	Comportamento	Partner	Data rate [kB/s]
Tastiera	Input	Umano	0,01
Mouse	Input	Umano	0,02
Stampante laser	Output	Umano	100 ÷ 10000
Floppy disk	I/O (memoria)	Macchina	50
Disco ottico	I/O (memoria)	Macchina	500
Disco magnetico	I/O (memoria)	Macchina	10.000
Rete-LAN	I/O	Macchina	20 ÷ 100.000
Video grafico (AGP)	Output	Umano	100.000

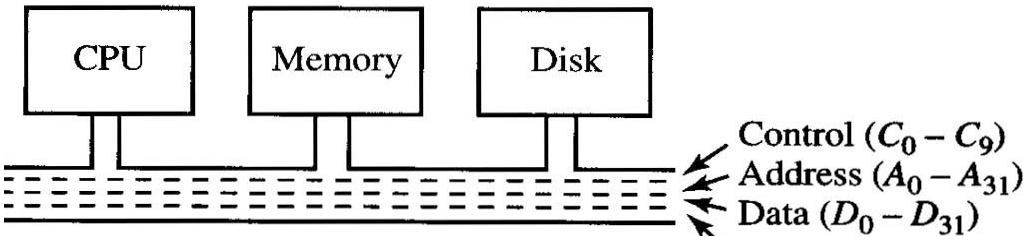
Collegamento: il **bus**

Come collegare CPU e periferiche? Connessione completa ? (tutti con tutti)

- N/2 *(N–1) connessioni bidirezionali: complesso e difficile da controllare



- **BUS:** pathway comune che connette i diversi dispositivi
  - Vantaggi bus: elevata flessibilità, semplicità, basso costo
  - Svantaggi: gestione complessa del canale condiviso
- Tipi 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)

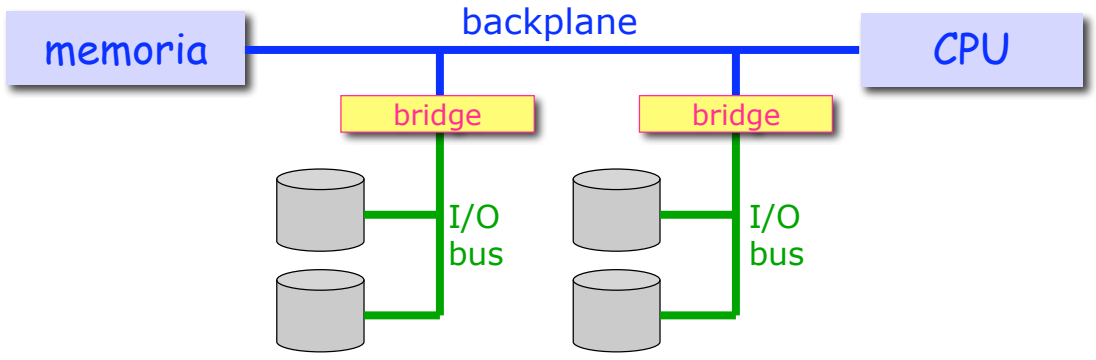


Tipologie di bus

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

Esempi:

- 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)

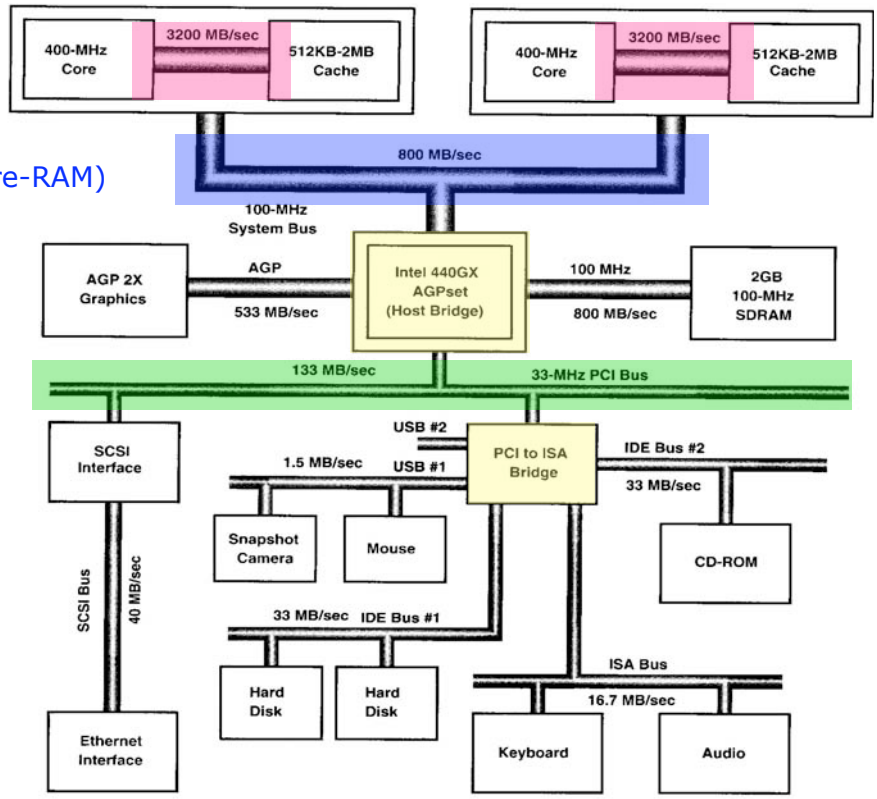


Esempio: **Pentium II – architettura PCI**

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

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

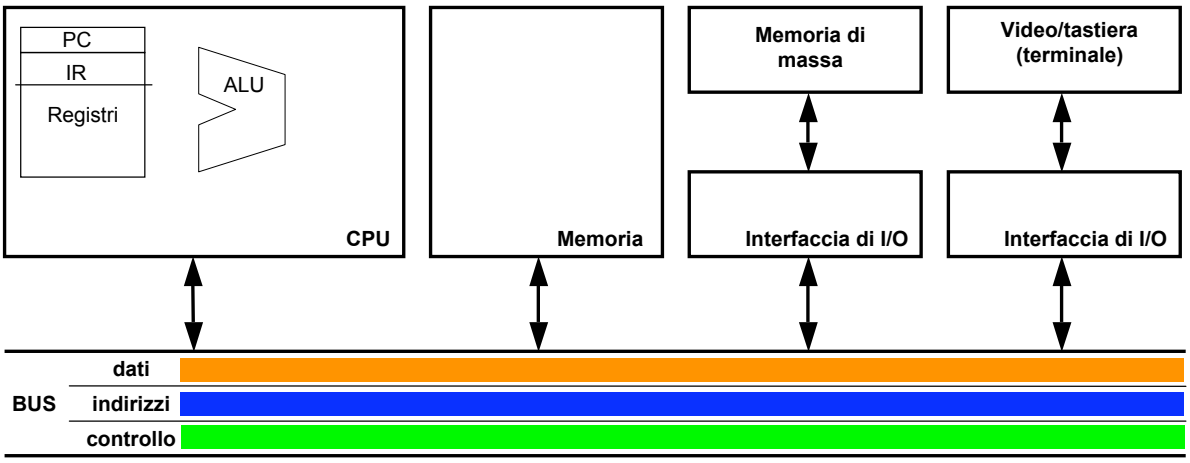
Bus di **backplane**: PCI (133 MB/s)



Esempio: bus di un elaboratore

- Suddivisione del bus in **parti logiche**:
  - **Bus DATI**
    - comprende le linee per trasferire **dati e istruzioni da/verso i dispositivi**.
    - In generale, la dimensione del bus dati è tale da garantire il trasferimento contemporaneo di una parola di memoria;
  - **Bus INDIRIZZI**
    - su cui la CPU provvede a trasmettere l'**indirizzo** da cui prelevare il dato nel caso di lettura dalla memoria, oppure in cui depositarlo nel caso di scrittura nella memoria
  - **Bus di CONTROLLO**
    - dove transitano le informazioni ausiliarie per la corretta definizione delle operazioni da compiere (es: comando di lettura o scrittura) e per la **sincronizzazione**
- Protocollo di comunicazione
  - La comunicazione su bus deve essere regolata attraverso un protocollo di comunicazione.

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 le diverse unità del calcolatore.*

  Generalmente composto da tre parti:
  - ***Bus dati***: le linee per trasferire dati e istruzioni da/verso i dispositivi.
  - ***Bus indirizzi***: su cui la *CPU* trasmette l'indirizzo di memoria da cui prelevare/depositare il dato nel caso di lettura/scrittura dalla memoria.
  - ***Bus di controllo***: informazioni ausiliarie per la corretta definizione delle operazioni da compiere e per la sincronizzazione tra *CPU* e memoria

- *Esempio: lettura dalla memoria*
  1. *La CPU fornisce l'indirizzo della parola desiderata sul bus indirizzi*
  2. *viene richiesta l'operazione di **lettura** attivando il bus di controllo.*
  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*

Problematiche di comunicazione su bus

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

Controllo di flusso:

I dispositivi collegati al bus hanno differenti velocità di reazione e di comunicazione

→ Solitamente all'interno dei dispositivi sono presenti **buffer** per mantenere l'informazione e non essere limitati dal dispositivo più lento.

Dimensionamento del buffer:

- Scelta ottima della dimensione di buffer:
  - Buffer **grossi** → Aumentare la **velocità di trasferimento**
  - Buffer **piccoli** → Ridurre i **tempi di risposta (latenza)**

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

Vantaggi

- I protocolli sincroni permettono di ottenere bus molto veloci

Svantaggi

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

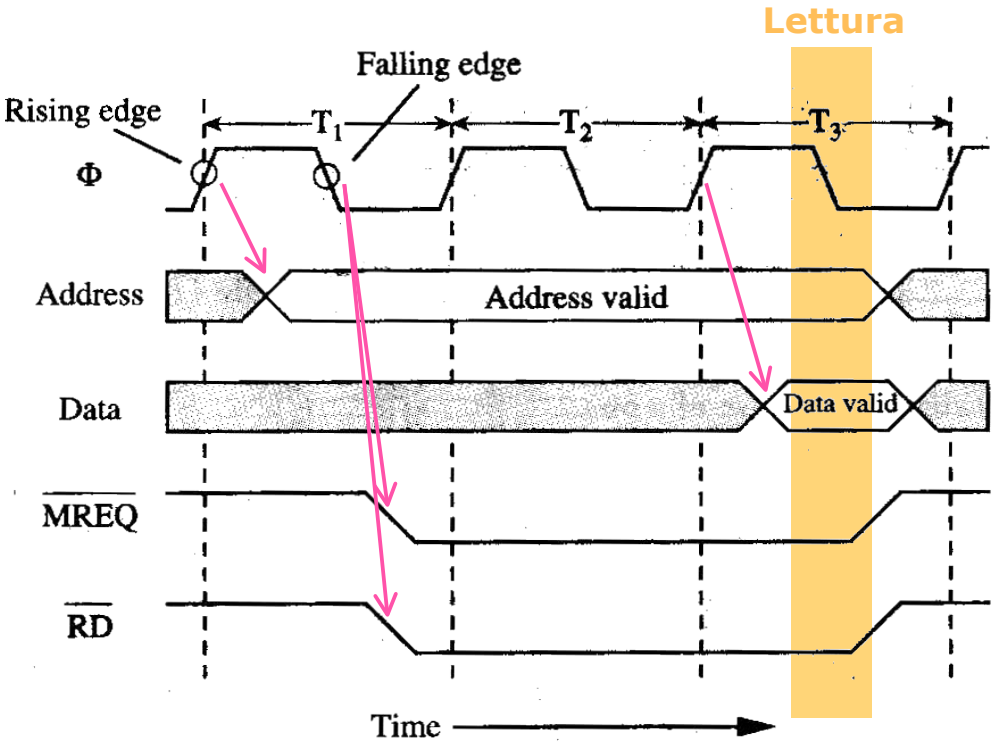
- I bus **processore-memoria** sono in genere **sincroni**:
  - hanno dimensioni ridotte, pochi elementi connessi e devono essere veloci
  - Ciclo di bus (**bus cycle**): n. cicli per effettuare una transazione: (typ. 2÷5 CC)

Bus sincroni

- Esempio bus sincrono: CPU – Memoria

Ciclo di **lettura memoria:**

Durata: 3 CC



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: Bassa velocità

- Esempio: comunicazione Master/Slave
  - Segnali: **`Master_Sync:`** `0: master ready, 1: master release`
    `Slave_Sync:` `0: slave ready, 1: slave release`

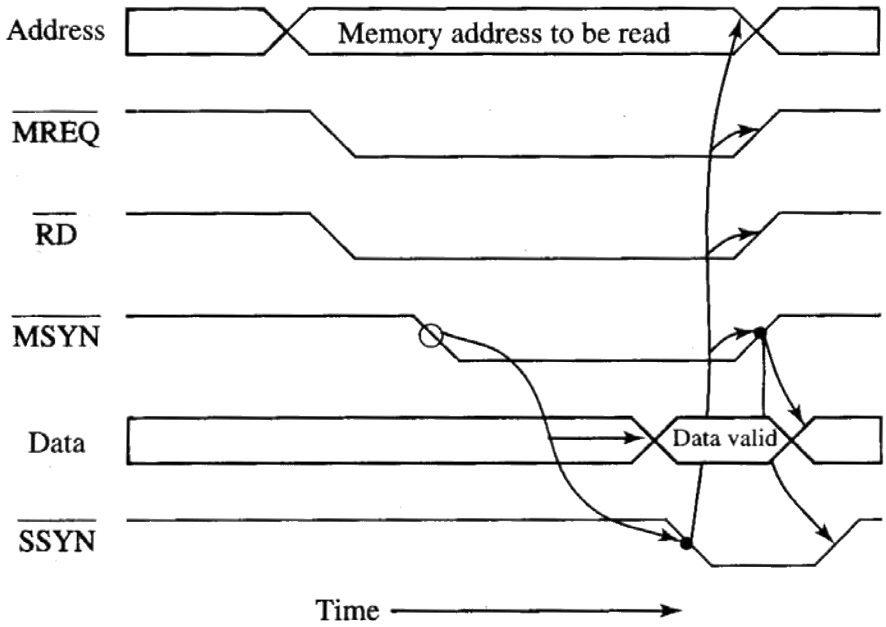
Protocollo di Handshaking:

(MSync→0) → (SSync→0) → ***COM*** *→ (MSync→1) → (SSync→1)*

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

Bus asincroni

- Esempio: lettura di memoria – comunicaz. **asincrona**



Alcuni tipi di bus...

	PCI	Firewire	SCSI
Nome dello standard	PCI	IEEE 1394	ANSI X3.131
Tipo di bus	Back-plane	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	0,5 metri (sincrono)	25 metri

Gestione del bus: arbitraggio

- E quando si hanno più di 2 dispositivi connessi al bus?
- L'architettura più semplice: **Master/Slave**
  - un unico **bus master** ("padrone" del BUS)
  - tutti gli altri dispositivi sono **slaves**

  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.
- Architetture con più dispositivi master:
  - occorre definire e rispettare una policy che coordini i diversi bus master
- **Arbitraggio del bus**
  - Questo è il principale inconveniente dei bus a nodo comune.

Protocolli di comunicazione: **arbitraggio**

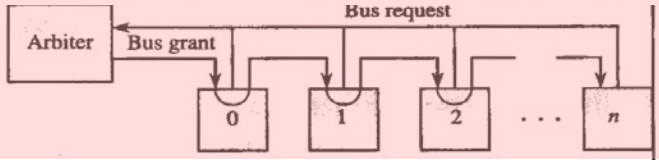
Arbitraggio:

- Occorre stabilire quale master 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**

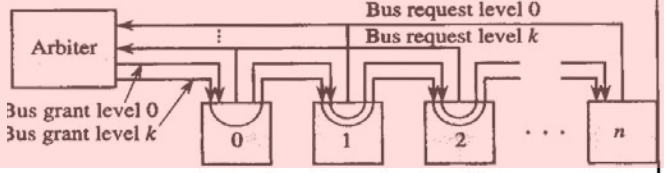
Schemi di arbitraggio

Schemi centralizzati

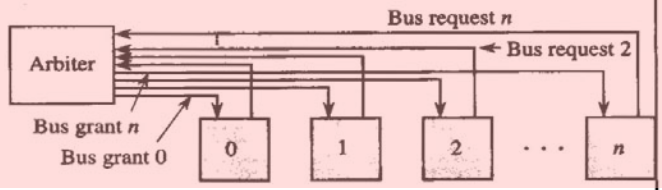
Arbitraggio Daisy Chain



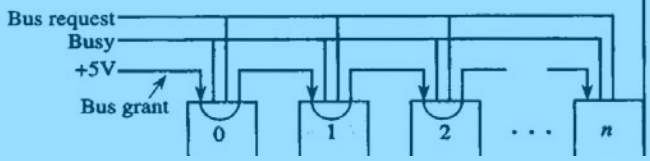
Arbitraggio Centralizzato con priorità



Arbitraggio Centralizzato Parallelo



Schema Decentralizzato (distribuito)



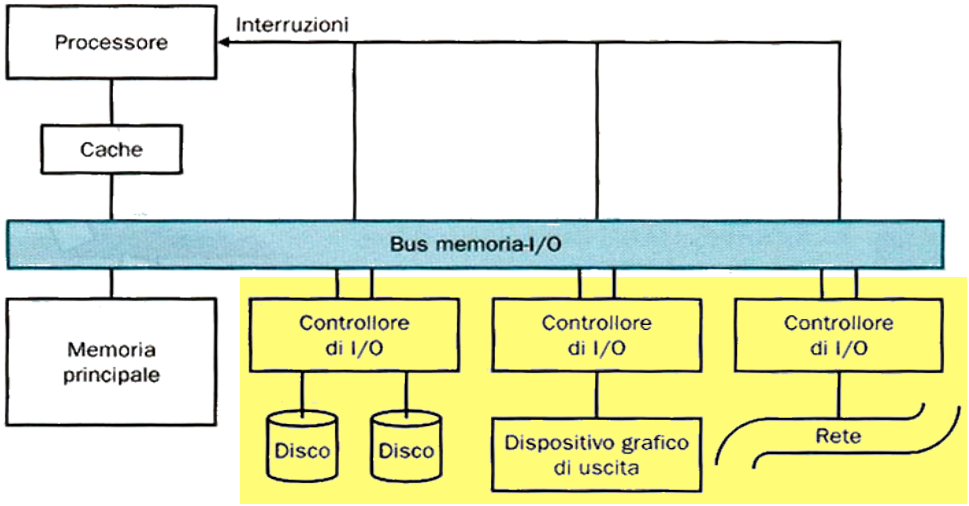
Arbitraggio distribuito

Un dispositivo che vuole prendere il controllo del bus, deve:

1. Inviare il segnale di richiesta del bus
2. Scrivere sul bus il codice che lo identifica
3. Controllare se il bus è libero (segnale **BUSY**)
4. Se il bus è libero ma ci sono richieste contemporanee di bus, controllare il codice dei dispositivi che hanno fatto richiesta
5. Se il bus è libero e i dispositivi hanno priorità minore:
   - inviare 0 sulla linea di bus grant ai dispositivi successivi, asserisce la linea di **BUSY** (il bus è occupato)
   - altrimenti non fare nulla
6. Deasserire la richiesta di bus

Le periferiche

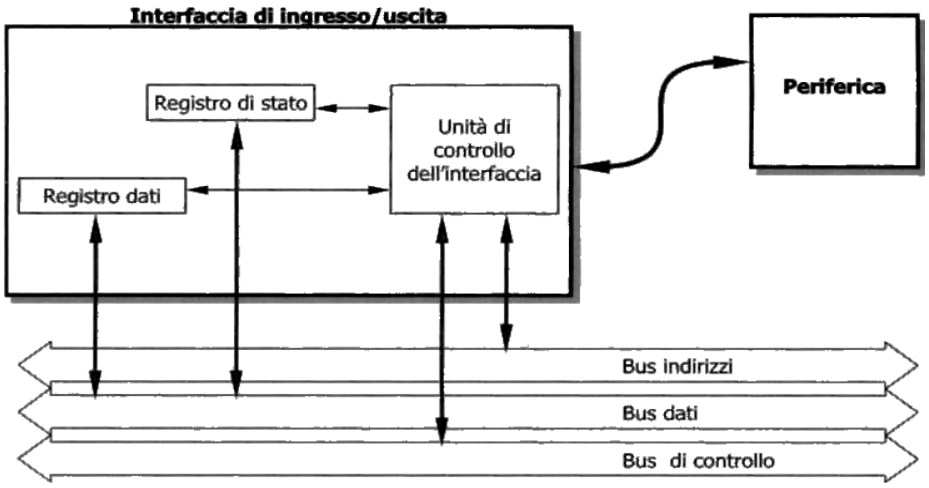
- PERIFERICHE: Dispositivi collegati alla CPU mediante bus e/o interfacce
- INTERFACCE:
  Metodiche standard (circuiti, protocolli) per la comunicazione.
  - Controllore della periferica (hardware)
  - Driver (software)



Interfacce: tassonomia

- 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** (Ethernet, Bluetooth, USB, …)
- Mezzo trasmissivo:
  - **Via cavo** Cavi in rame, Fibre ottiche
  - **Wireless** Infrarossi (IrDA),
    Onde radio (Bluetooth, WiFi, ZigBee)

Struttura di interfaccia di una periferica



- Solitamente 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 appartenente alla ISA che indirizzano direttamente il dispositivo
   - Nell'istruzione è contenuto:
     - Numero identificativo del dispositivo
     - Parola di comando (o indirizzo della parola di comando)

Indirizzamento **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

Indirizzamento Memory-mapped

- I controller 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`**