


# I bus

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

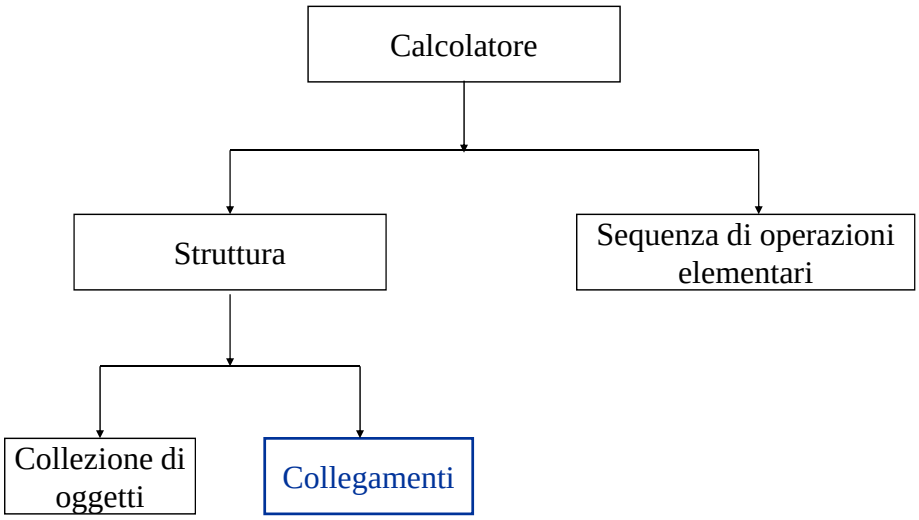
Università degli Studi di Milano

Riferimento Patterson: 8.1-8.3




# Sommario

# Descrizione di un elaboratore






# I/O

Dispositivi eterogenei per:

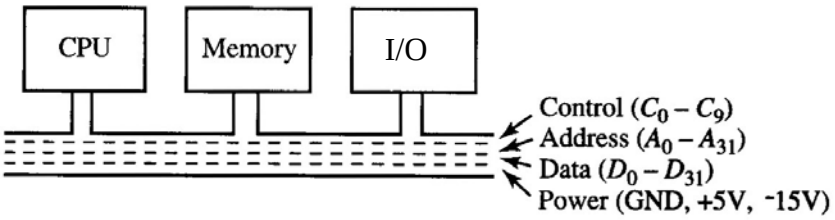
- velocità di trasferimento.
- latenze.
- sincronismi.
- modalità di interazione (con l'uomo o con una macchina)

# Dispositivi di I/O - classificazione

| Dispositivo | Comportamento | Partner | Tasso dati (KB/sec) |
|---|---|---|---|
| Tastiera | Input | Umano | 0.01 |
| Mouse | Input | Umano | 0.02 |
| Stampante laser | Output | Umano | 100.00 |
| Floppy disk | Memoria | Macchina | 50.00 |
| Disco ottico | Memoria | Macchina | 500.00 |
| Disco magnetico | Memoria | Macchina | 10,000.00 |
| Rete-LAN | Input o Output | Macchina | 20 – 1,000.00 |
| Videocamera (640x480@8bpp at 30Hz) | Input | Macchina | 9,216,000 |
| Video grafico (AGP) | Output | Umano | 100,000.00 |



# Il bus (connessione a cammino comune)



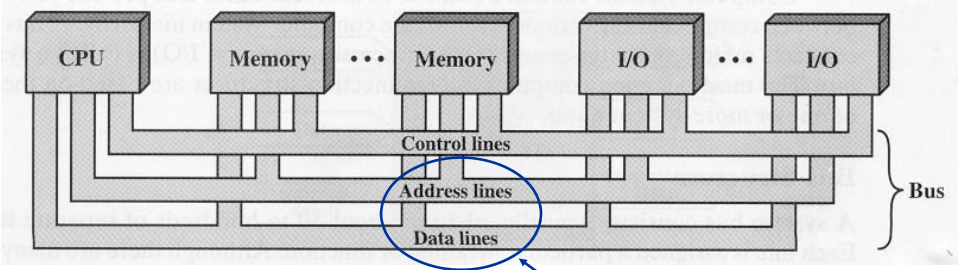

Pathway che connette tutti i dispositivi in modo bidirezionale (a partire dal PDP-8, omnibus, 1965).

Principali vantaggi della struttura a bus singolo:
- elevata flessibilità
- bassi costi.
- Problema: i dispositivi non possono trasmettere contemporaneamente.

Le architetture contengono uno o più bus che collegano questi tre componenti.

# La struttura del bus

3 gruppi funzionali: dati, indirizzi e segnali di controllo.



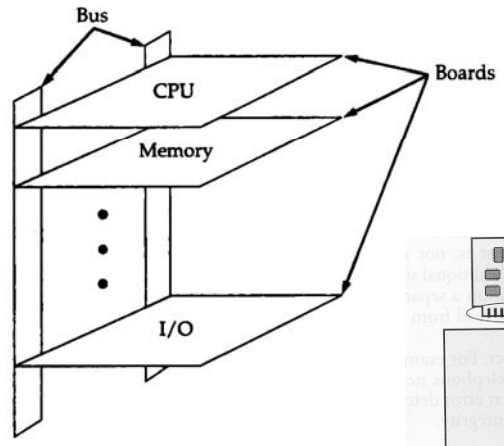

Fisicamente coincidenti nei bus recenti.

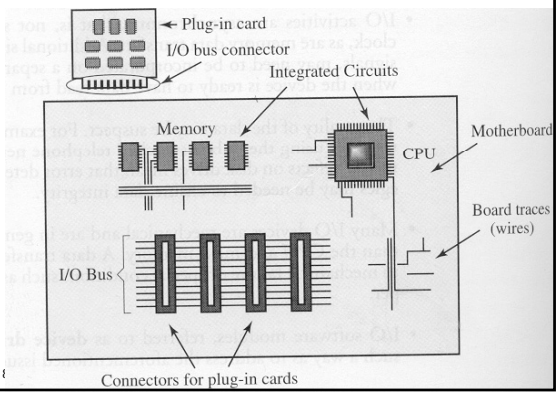
Data lines: ampiezza del bus = ampiezza della parola dati.
Address lines: capacità di indirizzamento (memoria principale + I/O).
Control lines: comandi e stato dei dispositivi.

Problema: quale dispositivo può trasferire sul bus in caso di richieste multiple? Chi decide?

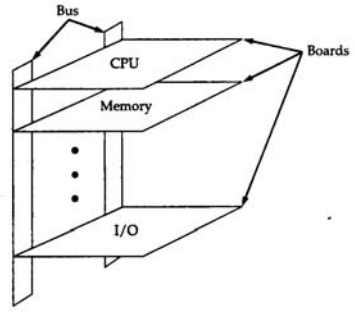


# Struttura fisica del bus

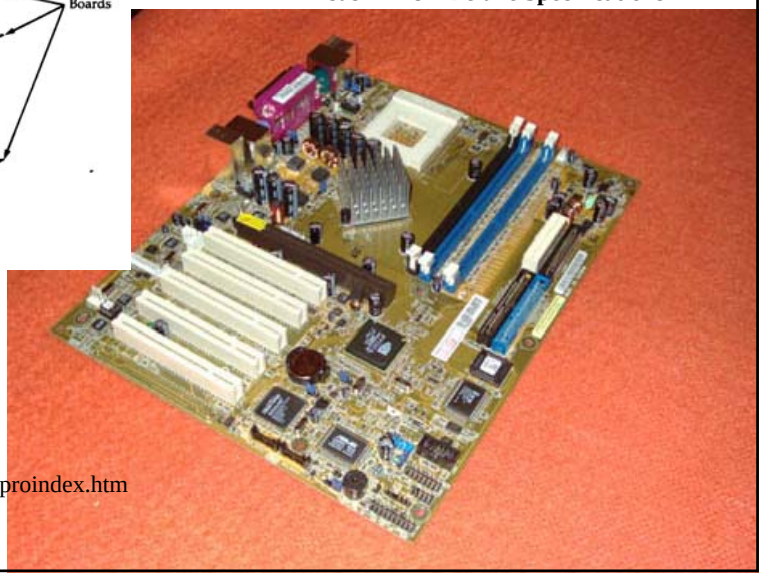


# Esempio di mother board



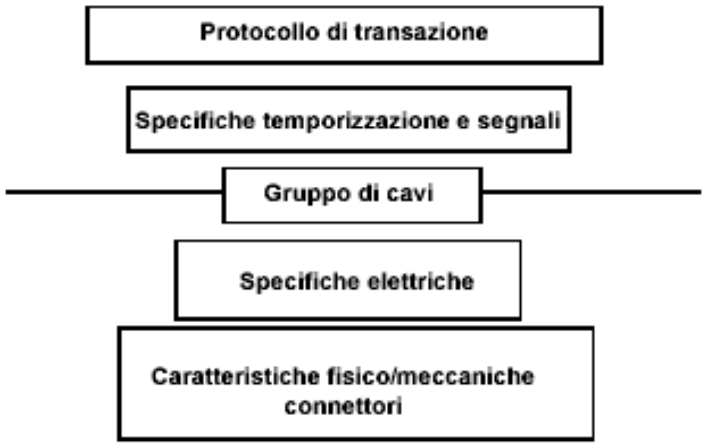

**Asus A7N8X Deluxe Specifications**



http://www.asus.it/products/proindex.htm

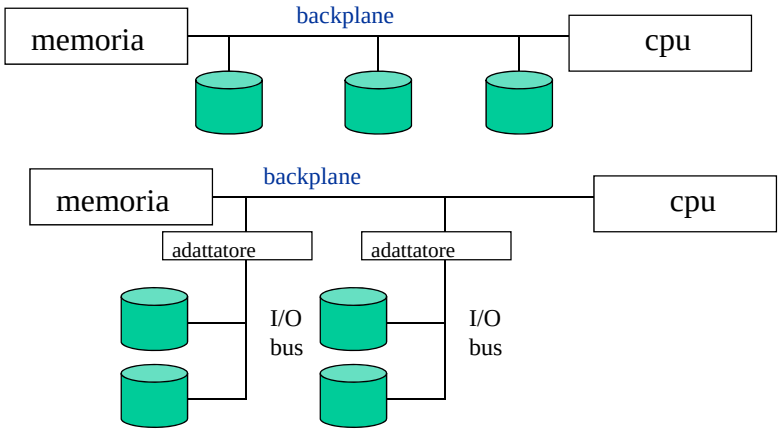


# Livelli di definizione



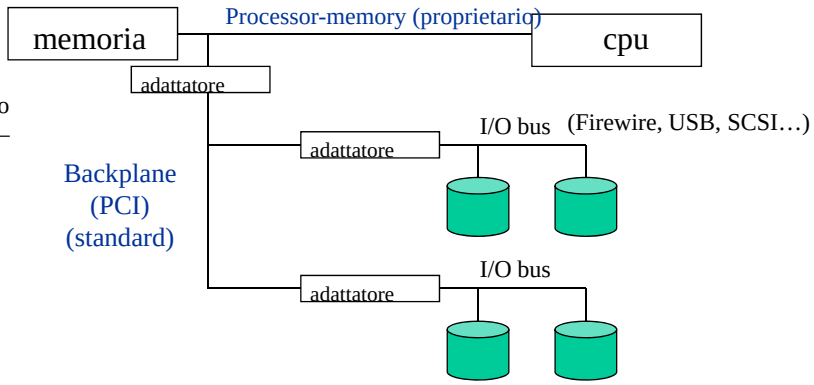

Transazione su bus: sequenza di operazioni che partono da una richiesta, e si concludono con il trasferimento di dati.

# Tipologie di bus – vecchio stile






# La gerarchia dei bus




Gli adattatori sono dispositivi attivi – bridges.

Processore-Memoria (cache): lunghezza ridotta, molto veloci, tipicamente sincroni.
Back-plane: notevole lunghezza, molti device connessi.
I/O: servono per far coesistere la memoria, il processore e i dispositivi di I/O su di un unico bus. Tipicamente asincroni.

Esistono due schemi principali di comunicazione su di un bus:
- Sincrono
- Asincrono


# I bus

| Caratteristica | PCI | SCSI |
|---|---|---|
| Tipo di bus | Back-plane | I/O |
| Ampiezza di base del bus (numero di segnali per i dati) | 32-64 | 8-32 |
| Numero di dispositivi master | molti | molti |
| Temporizzazione | Sincrono 33-66Mhz | Asincrono o sincrono (5-20Mhz) |
| Ampiezza di banda di picco teorica | 133-512MB/s (PCI64) | 5-40MB/s |
| Ampiezza di banda stimata raggiungibile per bus di base | 80MB/s | 2,5-40MB/s |
| Massimo numero di dispositivi | 1024 (32 dispositivi per segmento) | 7-31 |
| Massima lunghezza del bus | 0,5 metri | 25 metri |
| Nome dello standard | PCI | ANSI X3.131 |

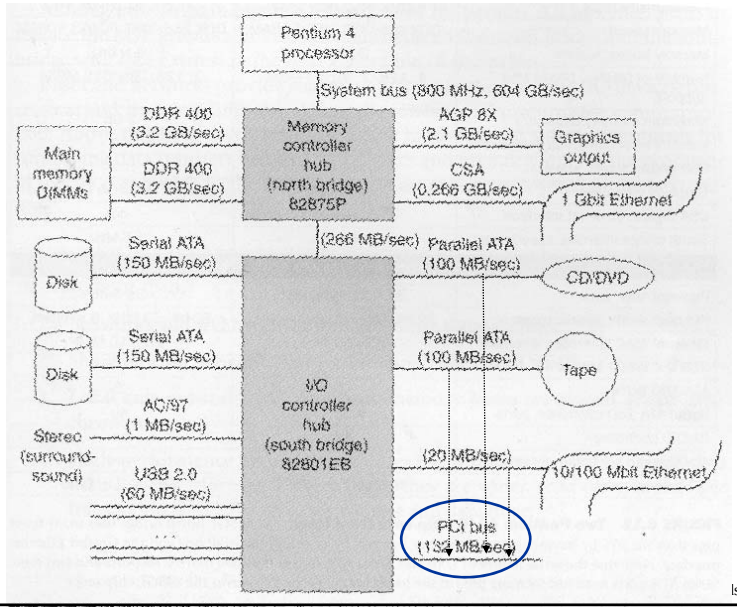



# Caratteristiche di bus asincroni

| Caratteristiche | Firewire (1394) | USB 2.0 |
|---|---|---|
| Tipo di BUS | I/O | I/O |
| Ampiezza bus (numero segnali) | 4 | 2 |
| Clock | Asincrono | Asincrono |
| Picco di velocità | 50Mbyte/s (Firewire 400)<br>100Mbyte/s (Firewire 800) | 0.2Mbye/s (low speed)<br>1.5Mbyte/s (full speed)<br>60Mbyte/s (high speed) |
| Numero massimo di device | 63 | 127 |
| Nome standard | IEEE 1394a, 1394b | USE Implementors Forum |
| Lunghezza massima | 4.5m | 5m |

Bus seriali ad alta velocità, punto a punto, con commutazione.

# Pentium IV






| | 875P chip set | 845GL chip set |
|---|---|---|
| Target segment | Performance PC | Value PC |
| System bus (64 bit) | 800/533 MHz | 400 MHz |
| **Memory controller hub ("north bridge")** | | |
| Package size, pins | 42.5 × 42.5 mm, 1005 | 37.5 × 37.5 mm, 760 |
| Memory speed | DDR 400/333/266 SDRAM | DDR 266/200, PC133 SDRAM |
| Memory buses, widths | 2 × 72 | 1 × 64 |
| Number of DIMMs, DRAM Mbit support | 4, 128/256/512 Mbits | 2, 128/256/512 MBits |
| Maximum memory capacity | 4 GB | 2 GB |
| Memory error correction available? | yes | no |
| AGP graphics bus, speed | yes, 8X or 4X | no |
| Graphics controller | external | Internal (Extreme Graphics) |
| CSA Gigabit Ethernet interface | yes | no |
| South bridge interface speed (8 bit) | 266 MHz | 266 MHz |
| **I/O controller hub ("south bridge")** | | |
| Package size, pins | 31 × 31 mm, 460 | 31 × 31 mm, 421 |
| PCI bus: width, speed, masters | 32-bit, 33 MHz, 6 masters | 32-bit, 33 MHz, 6 masters |
| Ethernet MAC controller, interface | 100/10 Mbit | 100/10 Mbit |
| USB 2.0 ports, controllers | 8, 4 | 6, 3 |
| ATA 100 ports | 2 | 2 |
| Serial ATA 150 controller, ports | yes, 2 | no |
| RAID 0 controller | yes | no |
| AC-97 audio controller, interface | yes | yes |
| I/O management | SMbus 2.0, GPIO | SMbus 2.0, GPIO |

I bridge

# Il PCI Express

| Caratteristica | PCI | PCI Express |
|---|---|---|
| Tipo di bus | Back-plane | Versatile (back-plane, I/O) |
| Ampiezza di base del bus (numero di segnali per i dati) | 32-64 | 4 + 4 (In/Out) |
| Numero di dispositivi master | molti | 1 |
| Temporizzazione | Sincrono 33-66Mhz | Sincronizato: 2.5GHz |
| Modalità di funzionamento | Sincrona parallela (1,024Mbit/s – 4,096Mbit/s) | Sincrona seriale (2,5Gbit/s) |
| Ampiezza di banda di picco teorica | 133-512MB/s (PCI64) | 300MB/s per direzione |
| Ampiezza di banda stimata raggiungibile per bus di base | 80MB/s | 300MB/s |
| Massimo numero di dispositivi | 1024 (32 dispositivi per segmento) | 1 |
| Massima lunghezza del bus | 0,5 metri | 0,5 metri |
| #Mbyte / s / linea | 4-16 | 75 + 75 |
| Nome dello standard | PCI | PCI - Express |







**Struttura gerarchica – gerarchia + stella**

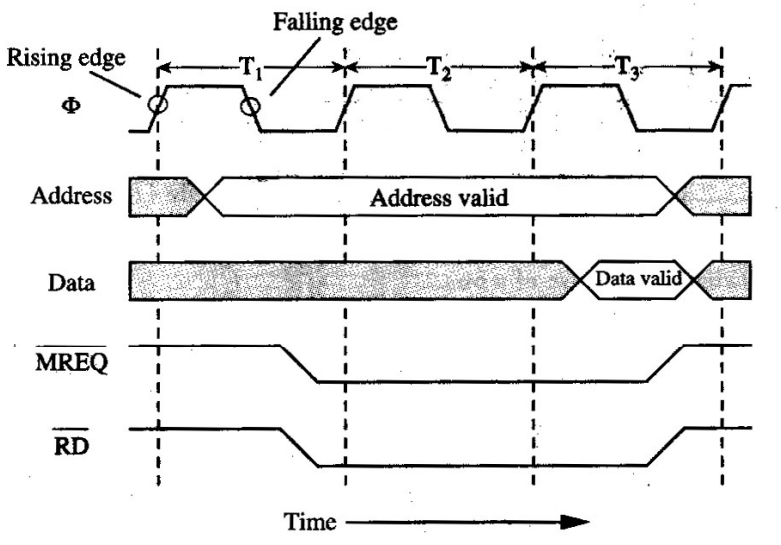
# Bus sincroni

- The le linee di controllo è presente la linea che porta il segnale di clock **(bus clock).** Esiste un protocollo di comunicazione scandito dai cicli di clock, in generale diverso (ma sicnronizzato) da quello della CPU.
- Questo tipo di protocollo permette di ottenere bus molto veloci.
- *Svantaggi:*
  - Ogni device deve essere sincronizzato.
  - Lunghezza limitata (per evitare che i ritardi nei fronti dovuti alla propagazione producano disallineamenti, clock skew).
  - Tutti i dispositivi devono potere lavorare alla frequenza imposta dal bus clock.
- I bus processor-memory sono spesso sincroni in quanto:
  - hanno  dimensioni ridotte.
  - hanno pochi elementi connessi.

Ciclo di bus (***bus cycle***): numero di cicli per effettuare una transazione: tipicamente da 2 a 5 cicli di bus clock.



# Bus sincroni: esempio



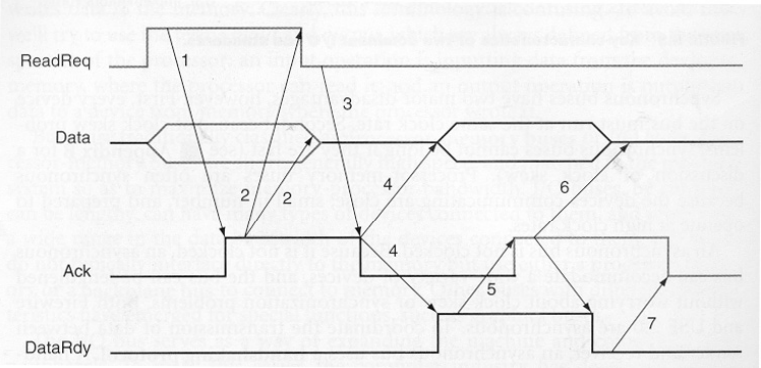
# Bus asincroni

- Un bus asincrono **non** è dotato di clock.
- La comunicazione tra due parti avviene mediante un protocollo di **handshaking.**
  **(!MSYN) -> Job -> (!SSYN) -> (MSYN) -> (SSYN)**
- I bus asincroni possono avere lunghezza elevata per connettere molti dispositivi.
- Sono efficienti quando i tempi di esecuzione delle varie periferiche variano molto tra loro.
- Spesso i bus di I/O sono asincroni.



# Esempio di handshaking

**ReadReq:** segnale di controllo di MemoryRead. In corrispondenza di questo segnale l'indirizzo della parola di memoria viene inviato sul bus dati.
**Data Rdy:** viene utilizzato per indicare che la parola è pronta sulle linee di dato. Questo segnale viene inviato dalla memoria quando il dato è disponibile in uscita dalla memoria.
**Ack:** viene utilizzato come risposta ai 2 segnali precedenti. Feed-back.




I segnali di ReadReq e DataReady rimangono alti fino a quando il ricevente (la memoria) non ha visto il comando e letto i dati corrispondenti.

# Sommario

I bus

La gestione dei bus

La gestione dell'I/O



# Arbitraggio del bus

**Protocollo** La comunicazione su bus deve essere regolata attraverso un **protocollo di comunicazione**.

Viene introdotto il concetto di **bus master (padrone del bus)**, il cui scopo è quello di controllare l'accesso al bus.

L'architettura più semplice è quella che prevede un unico bus master (il processore) in cui tutte le comunicazioni vengono mediate dal processore stesso.

Questo può creare un collo di bottiglia. Ad esempio nel caso di trasferimento di dati da I/O a memoria.

Si utilizza allora un'architettura con più dispositivi master.

In questo caso occorre definire e rispettare una policy che coordini i vari dispositivi bus master. Questa policy si chiama di **arbitraggio** del bus. Un solo dispositivo alla volta può essere master, tutti gli altri ascoltano.

Questo è il principale inconveniente dei bus a nodo comune.

# Protocollo di arbitraggio

Occorre stabilire quale master autorizzare all'utilizzo del bus.

Ad ogni dispositivo viene assegnata una *priorità*.

Il dispositivo a priorità maggiore può accedere prima al bus.

***Meccanismo di accesso al bus diventa:***
1. Richiesta del bus (*bus request*)
2. Assegnamento del bus (*bus grant*)

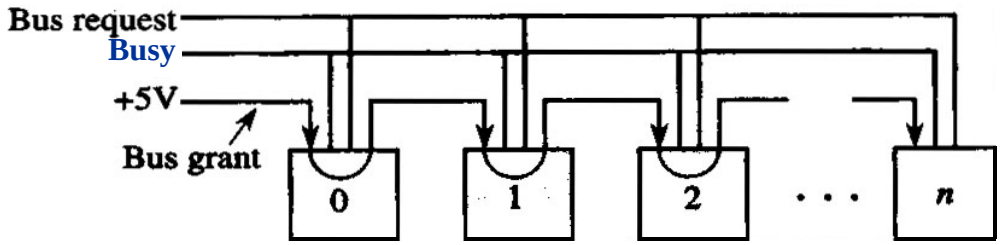
Problema è assicurare una *fairness*.

Compromesso tra fairness e priorità.

Un arbitro si preoccupa quindi di gestire *bus request* e *bus grant*.



# Arbitraggio distrubuito in Daisy Chain



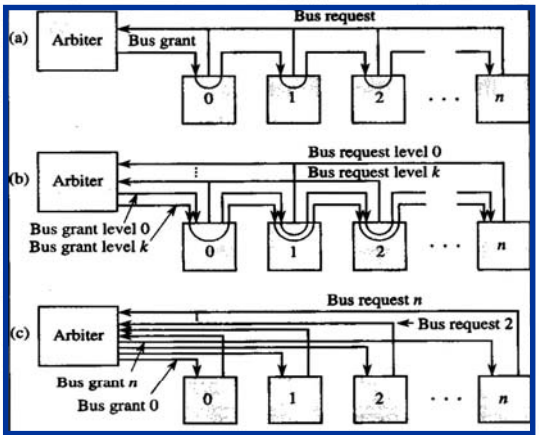

Viene introdotto il segnale di Busy.

# Schemi di arbitraggio centralizzati

Arbitraggio
In Daisy Chain

Arbitraggio
Centralizzato
Con priorità

Arbitraggio
Centralizzato
Parallelo




- Arbitraggio centralizzato parallelo non scala con il numero di dispositivi.
- Nell'arbitraggio centralizzato con priorità, un dispositivo elabora un solo segnale di grant, gli altri segnali di grant vengono fatti passare inalterati.



# Arbitraggio distribuito con autoselezione

In questo schema 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.
- 4) Se il bus non è occupato ma ci sono richieste contemporanee di bus, controllare il codice dei dispositivi che hanno fatto richiesta.
- 5) Occupare il bus se è libero o 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. *Ciascun dispositivo è arbitro.*
- 6) Deasserire la richiesta del bus.

Problema: rilevamento delle collisioni. Occorre prevedere un segnale di ricevuto.

# Sommario

I bus

La gestione dei bus

La gestione dell'I/O



# I/O

Dispositivi eterogenei per:

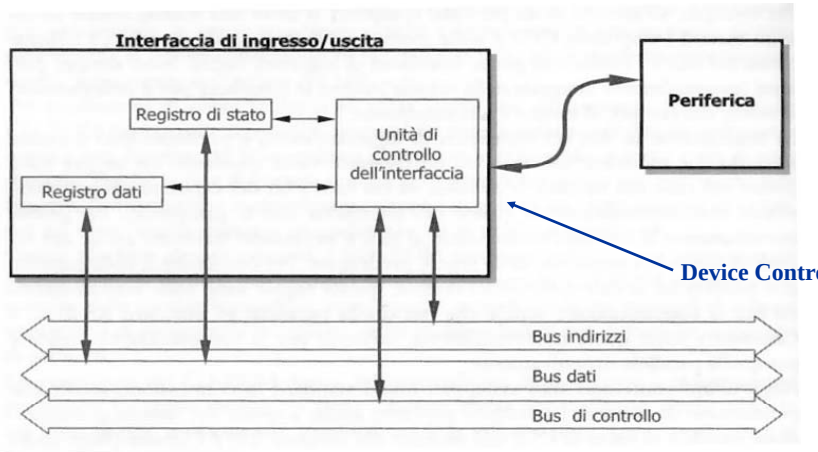
- velocità di trasferimento.
- latenze.
- sincronismi.
- modalità di interazione (con l'uomo o con una macchina)

# Bus & buffer

- I dispositivi sono collegati al bus tramite porte.
- I dispositivi collegati al bus variano in termini di velocità dell'esecuzione delle operazioni ⇒ necessario un meccanismo di sincronizzazione per garantire il trasferimento efficiente delle informazioni sul bus.
- Tipicamente all'interno delle unità che utilizzano il bus sono presenti dei **registri di buffer** per mantenere l'informazione durante i trasferimenti e non vincolarsi alla velocità del dispositivo più lento connesso al bus.
- All'interno dell'ampiezza di banda massima si può:
  - **Aumentare la velocità di trasferimento.** Buffer grossi.
  - **Ridurre i tempi di risposta (latenza).** Buffer piccoli.



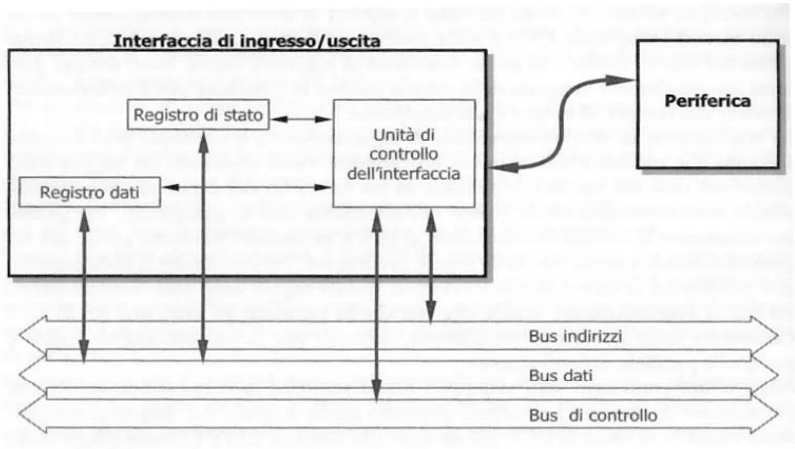
# Porta I/O




*Registri:*
•Dati
•Stato: situazione della periferica (idle, busy, down….) e comando in esecuzione.

Il driver agisce inviando alla periferica comandi opportuni e dati.

# Modalità trasferimento dati




Parallela (centronics). 1 byte alla volta.
Parallela (bus PCI, bus processore-memoria).
Seriale (RS232, RS432). 1 bit alla volta.
Nuovi standard sono USB, Firewire (IEEE 1394) e Bluetooth / 802.11 / 802.15 (wireless).



# Funzionamento di un driver

1. CPU richiede alla periferica (controller della periferica) l'esecuzione di un'operazione di read o di write.
2. I dati coinvolti nell'operazione devono essere trasferiti da e verso la memoria centrale.

Per potere eseguire un'operazione di read / write, occorre spesso una serie di operazioni sul dispositivo, che vengono eseguite attraverso il controller.

Esempio: una stampante ha 1 registro dati ed 1 registro di stato. Il registro di stato contiene il bit done, che viene impostato a 1 quando il carattere è stato stampato; ed il bit error che, indica se ci sono problemi. Il processore deve controllare che non ci siano errori e che il bit di done sia stato settato ad 1 prima di inviare un altro dato.

Per potere inviare i comandi al controller, occorre prima avere individuato il controller giusto!!
A ciascun dispositivo viene dato uno o più numeri -> indirizzo personalizzato.

2 modalità:
- Memory-mapped
- Istruzioni speciali di I/O

# Istruzioni speciali di I/O

Istruzioni appartenente alla ISA che indirizzano direttamente il dispositivo (i registri del dispositivo):
- Numero del dispositivo
- Parola di comando (o indirizzo della parola che contiene il comando)

Sul bus è possibile inviare il numero del dispositivo su un insieme di linee dedicate.
Ci saranno linee dedicate anche ai segnali di controllo (read / write).
I dati viaggeranno sulle linee dedicate ai dati.

Rendendo le istruzioni illegali al di fuori del kernel mode del processore, i programmi utenti non accedono direttamente ai device controller.

Esempio di architetture di questo tipo: Intel IA-32, IBM370.



# 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 l'*indirizzo* è al di fuori dallo spazio fisico della memoria.

• 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.

• Gli indirizzi riservati ai registri del controller fanno di solito riferimento alla porzione di memoria riservata al SO e non accessibile quindi al programma utente.

• I programmi utente devono quindi passare dal SO per accedere a questi indirizzi riservati (**modalità kernel**) e quindi effettuare operazioni di I/O. Questo è quanto viene fatto ricorrendo alle System Call.

# Esempio: Receiver (tastiera)

```
# NB i dispositivi vengono indirizzati tramite gli indirizzi "alti".

.text
.globl main
main:
        li $t0, 0x8000 0000 # indirizzo del receiver control register (2Gbyte)
        li $t2, 0x8000 0004  # indirizzo del receiver data register

# Ciclo di lettura di un carattere
ciclo:  lw $t1, 0($t0)          # Contenuto del registro di controllo
        andi $t1, $t1, 0x1      # if $t1 = = 1 esci
        beq $t1, $zero, ciclo

        lw $a0, 0($t2)          # Caricamento del dato in a0

        li $v0, 10              # exit
        syscall
```



# Sommario

Il bus ed il protocollo di trasferimento

Tipologie di bus

La gestione dell'I/O