



**La Sapienza**

Università degli Studi di Roma

Dipartimento di Informatica e Sistemistica

# CALCOLATORI ELETTRONICI

## BUS ed operazioni di I/O

**Emiliano Trevisani**

**[trevisani@dis.uniroma1.it](mailto:trevisani@dis.uniroma1.it)**

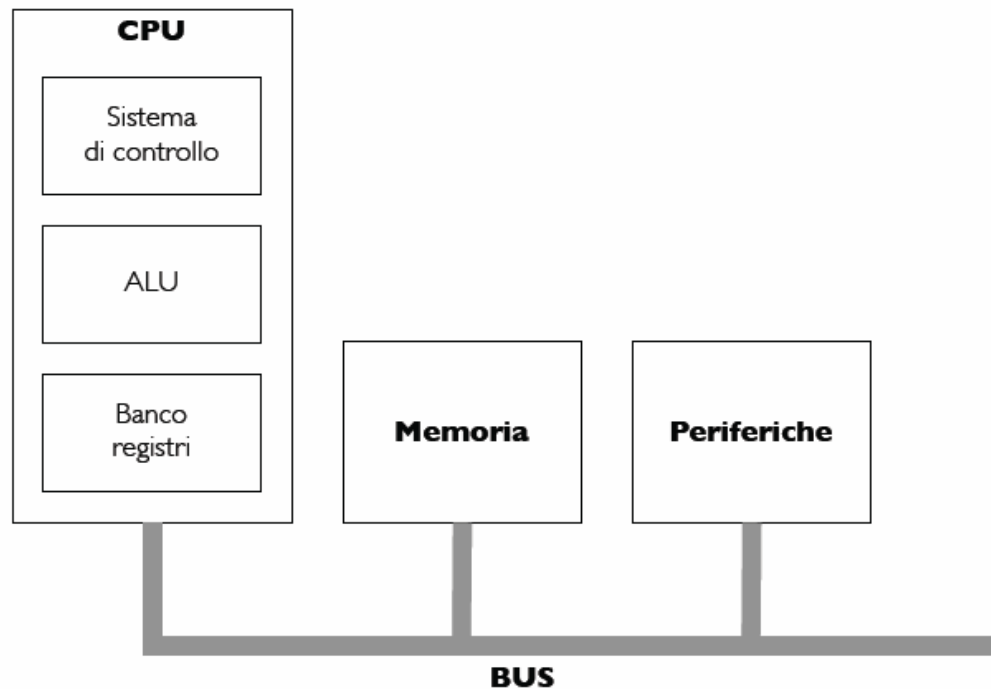
**A.A. 2007/2008**

# Bus



□ Con riferimento al modello di Von Neumann già discusso:

- il sistema “calcolatore” può essere definito come un insieme di unità “funzionali” [CPU, sistema di Memoria, periferiche di I/O] interconnesse da una **struttura di interconnessione [BUS]**
- Un bus è generalmente costituito da un insieme di collegamenti elettrici
- Alcuni parametri caratterizzanti un bus: **larghezza [#bit], clock, arbitraggio**



# Bus



- In un calcolatore moderno esistono, in realtà, **bus multipli**
  - bus interni
    - confinati all'interno di una singola unità funzionale;
    - interconnettono i sottosistemi dell'unità;
    - tipicamente gestiti con meccanismi “proprietary”
  - bus esterni:
    - si estendono all'esterno di un'unità funzionale per interconnetterla ad altre unità funzionali.
    - tipicamente gestiti con meccanismi standardizzati.
  - Soddisfano diverse esigenze:
    - velocità di trasferimento
    - diverse “larghezze”
    - più trasferimenti paralleli
    - compatibilità “all'indietro”

## Bus

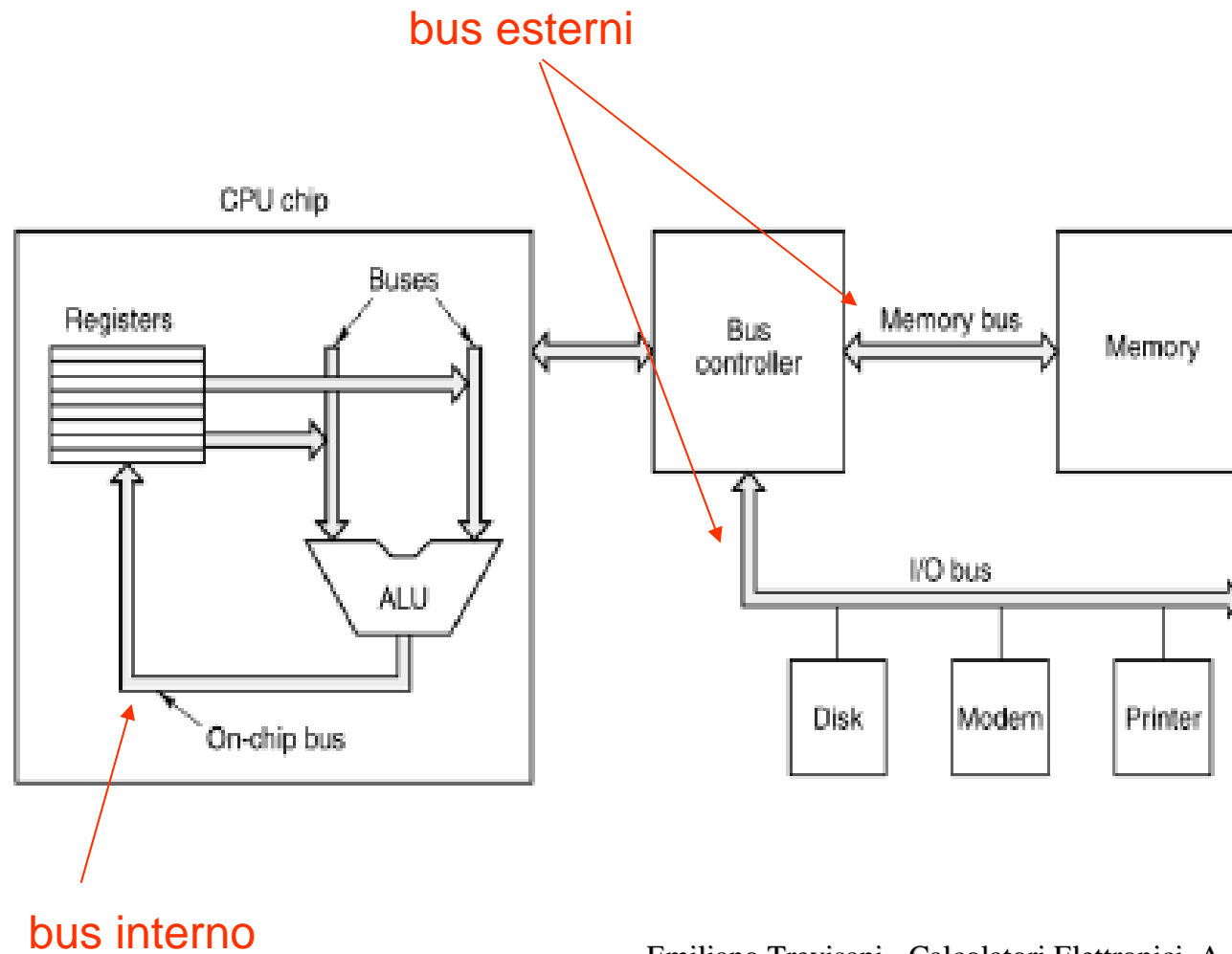


- ❑ I primi calcolatori erano dotati di un unico BUS esterno (**BUS di sistema**), collegante CPU, memoria e unità di I/O
- ❑ La maggior parte dei calcolatori moderni è dotata di due bus esterni: **BUS di memoria** (CPU ↔ sistema di memoria) e **BUS di I/O** (CPU ↔ periferiche di I/O)
- ❑ Il numero dei BUS esterni può anche essere superiore a 2

# Bus



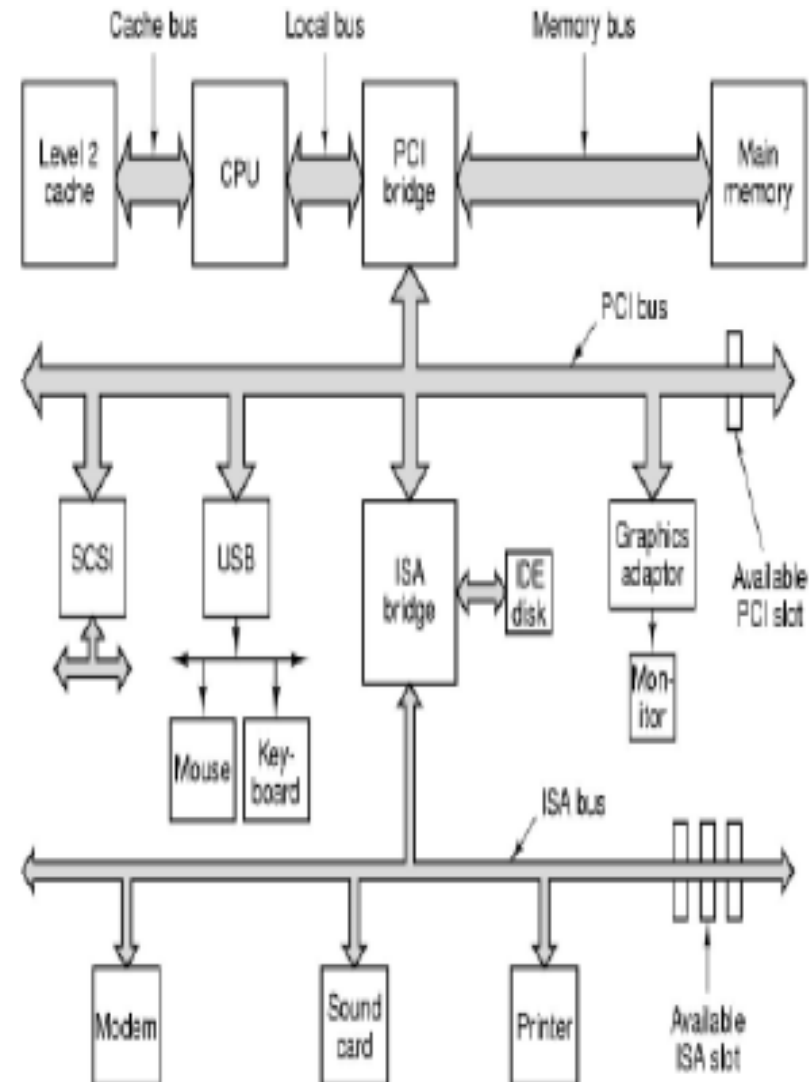
## □ Esempio



# Bus



□ Esempio: CPU Pentium II



# Bus



## □ Gestione di BUS condiviso

- Controllo del Bus: quali unità usano in un certo intervallo di tempo un Bus; in particolare: come vengono gestite le esigenze d'accesso simultanee da parte di diverse unità [“conflitti”]
- Funzionamento del Bus: si riferisce all'esecuzione delle operazioni di I/O sul Bus (“temporizzazione”)
- Modalità di I/O: chi “esegue” le operazioni di I/O fatte sul BUS

## □ Controllo del BUS

- In ogni istante, una sola unità possiede il controllo del BUS decidendo quali operazioni di I/O eseguire
- In genere la CPU possiede il controllo del BUS (o dei BUS) ma può anche cedere temporaneamente questo ruolo ad altre unità [DMA]
- L'unità che detiene il controllo del BUS si dice **MASTER**; le restanti unità funzionali si dicono **SLAVE**
- Ogni BUS è controllato da un unico MASTER che anche può controllare più BUS. Il calcolatore deve possedere almeno un MASTER (di solito la CPU)
- Ogni BUS può collegare più SLAVE

## Bus



### □ Arbitraggio del BUS

- La CPU ha generalmente il ruolo di MASTER tra le unità funzionali
- Tuttavia, in circostanze specifiche ed in accordo a criteri di efficienza, anche altre unità funzionali possono assumere temporaneamente il ruolo di MASTER. Ad esempio, una periferica di I/O può diventare temporaneamente MASTER per trasferire dati direttamente in la memoria, senza il controllo diretto da parte della CPU (Modalità DMA).
- In questi casi [cessione del ruolo di MASTER da un'unità funzionale ad un'altra], occorre un meccanismo di **ARBITRAGGIO DEL BUS** per gestire richieste simultanee di accesso al Bus stesso
- Il meccanismo di arbitraggio di un BUS ne regola l'utilizzo da parte delle diverse unità funzionali su di esso attestate
- Esistono due meccanismi principali di arbitraggio:
  - **centralizzato**
  - **distribuito**

## Bus



### □ Arbitraggio centralizzato

- Un'unità funzionale dedicata svolge la funzione di **arbitro del BUS**
- Alcune linee di controllo collegano l'arbitro del BUS alle unità funzionali che potrebbero richiedere il controllo del BUS
- L'arbitro implementa il meccanismo di cessione del controllo del BUS (cessione del ruolo di MASTER)

### ▪ Arbitraggio distribuito

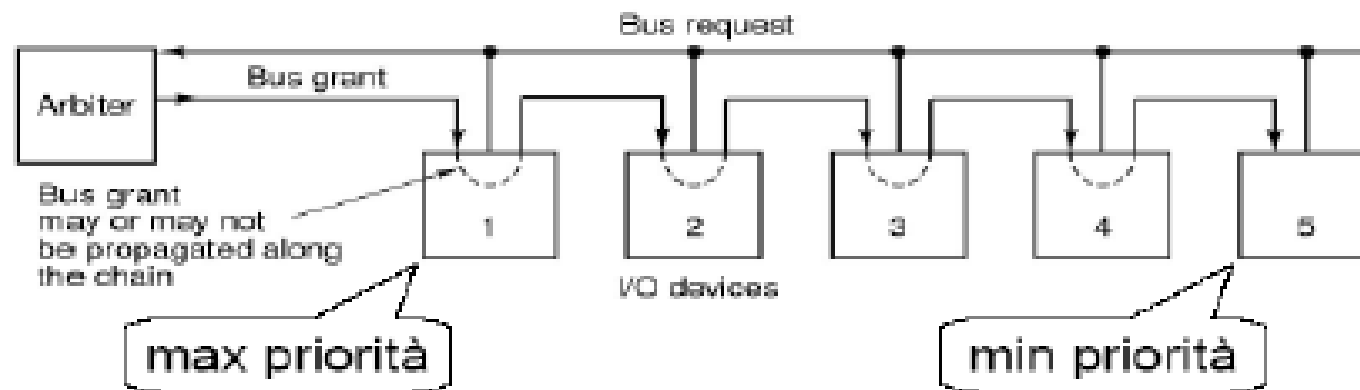
- Non prevede l'utilizzo di un unità centralizzata come arbitro del BUS
- Il meccanismo di arbitraggio è **distribuito** tra le varie unità, ciascuna delle quali contribuisce ad implementare il protocollo di gestione
- Il meccanismo utilizza delle linee di controllo per la comunicazione fra le unità funzionali

## Bus



### □ Esempio: arbitraggio centralizzato in “Daisy Chain” [festone]

- Quando l'arbitro riceve una richiesta di BUS (BUS request), attiva la linea di conferma (BUS grant).
- La conferma viene passata in cascata alle unità potenziali richiedenti in accordo alle seguenti considerazioni:
  - se un'unità non ha una richiesta pendente, passa la conferma all'unità successiva
  - se un'unità ha una richiesta pendente trattiene per sé la conferma, senza passarla all'unità successiva, assumendo il controllo del BUS [diventa MASTER ]
- Le unità più “lontane” dall'arbitro risultano sfavorite

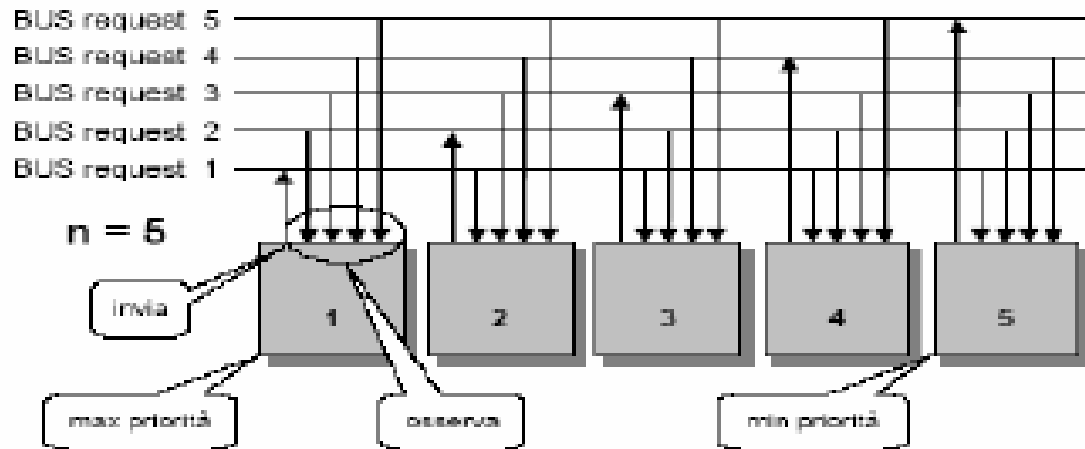


# Bus



## □ Esempio: arbitraggio distribuito ad “n linee”

- Prevede una linea di richiesta BUS per ogni unità (n linee); non esiste alcun arbitro del BUS; le unità hanno priorità diverse e prefissate.
- Ciascuna unità osserva **tutte** le linee di richiesta BUS; ogni unità può attivare solo la **propria** linea di richiesta BUS
- Quando l'unità vuole diventare MASTER, verifica se almeno un'altra unità con priorità superiore alla sua ha attivato la propria linea di richiesta BUS
  - In caso negativo, l'unità attiva la sua linea di richiesta BUS e ottiene il controllo del BUS, diventando MASTER
  - In caso positivo, ritarda la richiesta
- OSS: si evita l'unità arbitro, ma il numero di linee di controllo cresce con quello delle unità

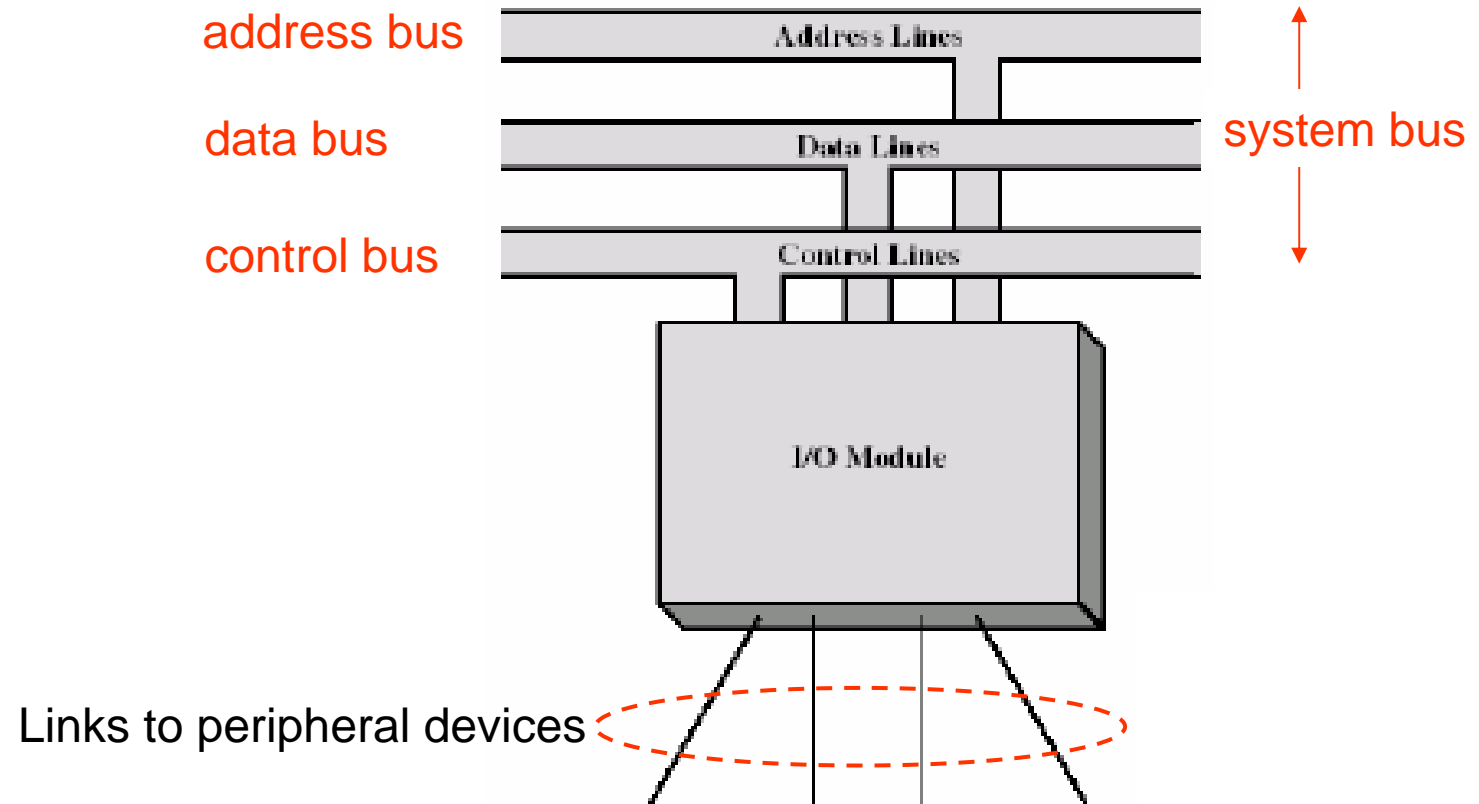




- ❑ In generale, le periferiche di I/O non vengono connesse direttamente al bus di sistema ma attraverso un modulo dedicato che agisce da **interfaccia** con CPU e memoria [**modulo di I/O**]
  - Le periferiche sono molto diverse tra loro in termini di:
    - Funzionalità: es. stampante, modem, dischi magnetici,...
    - Velocità di trasferimento dati [comunque inferiori a CPU e memoria]
    - Formato e lunghezza dei dati
  - Queste considerazioni giustificano l'adozione dei moduli di I/O d'interfaccia tra CPU, memoria e periferiche [es. sarebbe necessario adottare una logica diversa per ogni formato]
  - La periferica deve gestire unicamente il dialogo con il controllore di I/O [modulo]; sarà quest'ultimo a gestire il dialogo con CPU e memoria
  - Esempio: la CPU dialoga con il modulo di I/O per la lettura di un dato da periferica

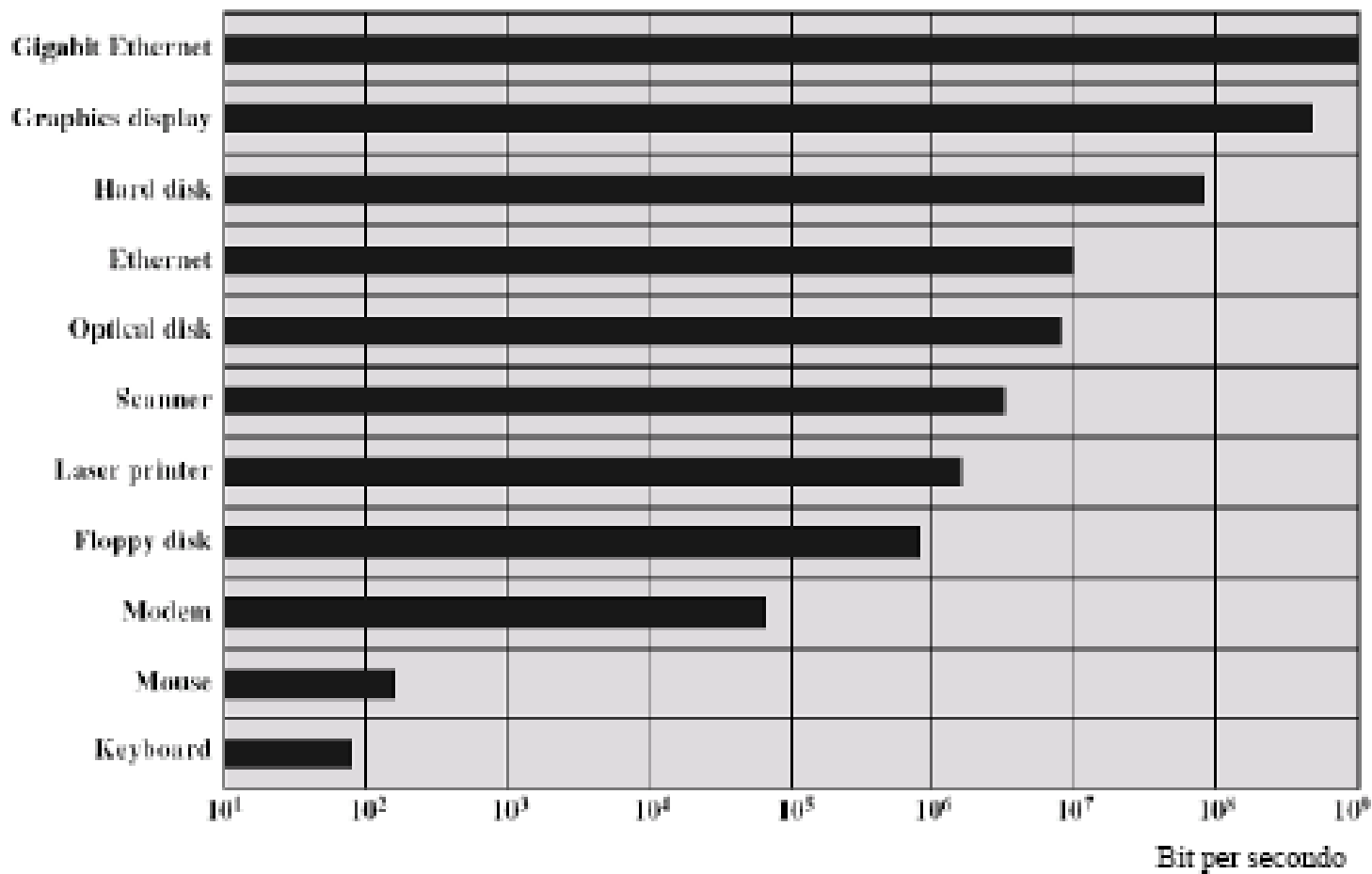
## □ Modulo di I/O

- controlla una o più periferiche
- gestisce il trasferimento dati da/verso la periferica



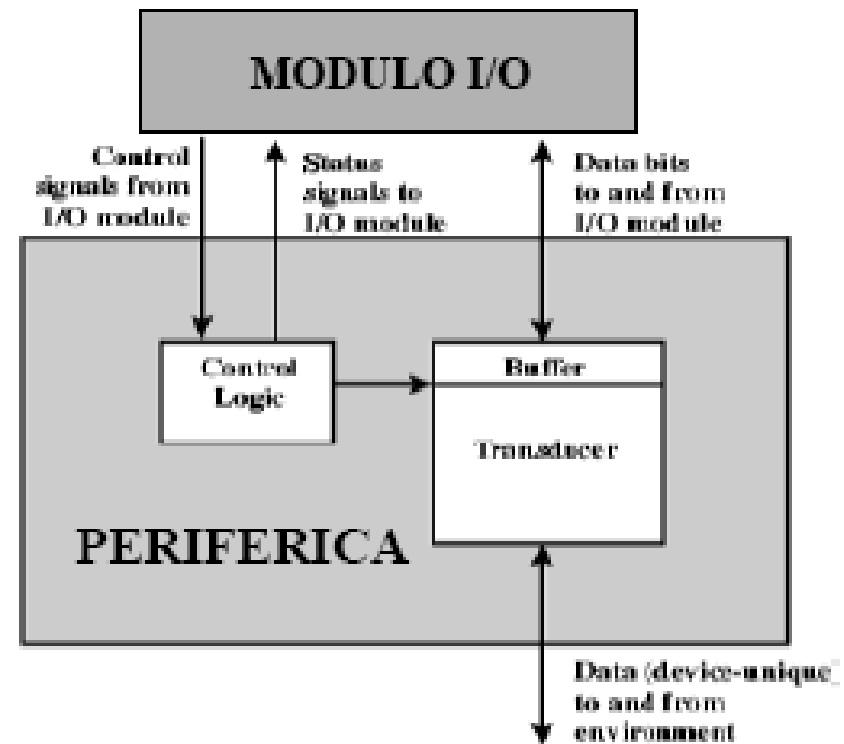


- Può essere interessante confrontare le velocità di trasferimento dati di alcune periferiche comuni:



## □ Modello generale di periferica

- Segnali di controllo [es. READ, WRITE,...]
- Segnali di stato [es. READY, BUSY,...]
- Dati da/verso periferica
- Logica di controllo
  - controlla il trasduttore - attuatore per l'esecuzione del comando esterno genera le variabili di stato
- Buffer





## □ Modulo di I/O

- Nasconde al sistema CPU / Memoria le caratteristiche elettromeccaniche della periferica; in questo modo i trasferimenti da/verso periferica sono gestiti come semplici operazioni di lettura/scrittura [es. da parte della CPU]
- Sulla base della complessità i moduli di I/O si distinguono in:
  - I/O controller: bassa complessità, richiedono un controllo estensivo da parte del processore; usati nei microprocessori di basse prestazioni
  - I/O processor: alta complessità, eseguono gran parte del controllo delle operazioni di I/O altrimenti richiesto al processore; usati nei microprocessori di alte prestazioni



## □ Tecniche di I/O

- Si distinguono tre modalità di gestione delle operazioni di I/O da/verso periferica; ciascuna implica un livello di complessità diverso per il modulo di I/O
  - **I/O programmato [Programmed I/O]:** controllo completo delle operazioni di I/O da parte del processore che esegue specifiche istruzioni di I/O
  - **Pilotato da interruzioni [Interrupt driven]** un segnale [**interrupt**] interrompe l'esecuzione di un programma per avviare un'opportuna “procedura” di gestione delle operazioni di I/O
  - **Accesso Diretto alla Memoria (“Direct Memory Access”, DMA)** un modulo dedicato [distinto dalla CPU] che ha accesso alla memoria controlla ed esegue le operazioni di I/O; questa modalità utilizza in genere anche i segnali d'interruzione

## □ Tecniche di I/O: overview

	<b>No Interrupts</b>	<b>Use of Interrupts</b>
<b>I/O-to-memory transfer through processor</b>	Programmed I/O	Interrupt-driven I/O
<b>Direct I/O-to-memory transfer</b>		Direct memory access (DMA)



## □ I/O programmato [Programmed I/O]

- La CPU controlla ed esegue completamente l'operazione di I/O
- Il processore invia/riceve dati direttamente dal modulo I/O controllando periodicamente lo stato della periferica (**polling**)
- Il processore attende che la periferica completi il task (es. lettura, scrittura) prima di proseguire l'esecuzione del programma
  - Inefficienza: CPU rallentata dai tempi della periferica, potrebbe proseguire nell'esecuzione introducendo parallelismo rispetto all'accesso al dato
- Il processore invia la richiesta di operazione di I/O all'**indirizzo** della periferica.
- Le istruzioni utilizzate per il trasferimento dati dai registri alla memoria vengono utilizzate come istruzioni I/O



## □ I/O programmato [Programmed I/O]

- Come indirizzare la periferica?
  - **“memory-mapped I/O”** – periferica **“mappata”** in memoria: alla periferica sono associati uno o più indirizzi di memoria. Un’operazione di lettura / scrittura su questi indirizzi in realtà si traduce in un accesso ad un registro della periferica
  - **“isolated I/O”**: il formato dell’istruzione è modificato in modo da distinguere indirizzi di memoria e indirizzi di periferica



## □ I/O pilotato da interruzioni [Interrupt driven I/O]

- Idea: nell'attesa che la periferica risponda, il processore può fare altro lavoro utile invece che interrogarne ciclicamente lo stato
- Quando la periferica è pronta, avverte il processore con un segnale sul bus di controllo (segnale di “interruzione” - interrupt) [**gestione asincrona**]
  - Appena possibile, il processore interrompe le sue attività eseguendo la **procedura di gestione dell'interruzione**
  - In genere, la CPU verifica la presenza di richiesta d'interruzione in punti fissi del ciclo istruzione
- La gestione è asincrona ma è sempre la CPU ad eseguire le operazioni di I/O durante la fase di gestione dell'interruzione

❑ **I/O pilotato da interruzioni [Interrupt driven I/O]**

- Procedura di gestione dell'interruzione:
  1. L'interfaccia della periferica genera l'interruzione [IRQ – Interrupt Request; diverse modalità e politiche di gestione: es. “daisy chain”]
  2. Nel momento in cui la CPU accetta l'interruzione l'interfaccia della periferica riceve il segnale di ACK
  3. La periferica invia sul bus il proprio vettore di interrupt; questo consente alla CPU di individuare l'indirizzo di memoria dell'appropriata routine di gestione
  4. La CPU “salva” il suo stato interno nello stack: registri PC e SR + registri generici
  5. La CPU inizia ad eseguire la routine di gestione caricando il suo indirizzo di memoria in PC
  6. La CPU “serve l'interruzione” completando l'esecuzione della corrispondente routine di gestione
  7. Prima di riprendere l'esecuzione nel punto d'interruzione, viene ripristinato lo stato interno della CPU utilizzando lo stack.



## □ Direct Memory Access (DMA)

- Limitazioni dell' I/O programmato e del I/O pilotato da interruzioni:
  - la velocità di trasferimento è limitata dalla velocità con cui il processore può “verificare” e “servire” una periferica
  - le istruzioni di I/O sono comunque eseguite dal processore
- Il modulo per l'Accesso Diretto alla Memoria (DMA) supera tali limiti, soprattutto quando il numero di trasferimenti è elevato
- Un DMA controller:
  - è un modulo hardware aggiuntivo al modulo di I/O
  - richiede in Input:
    - la locazione di memoria iniziale su/da cui trasferire i dati
    - il numero di parole di memoria da trasferire
    - l'indirizzo del modulo di I/O coinvolto



## □ Direct Memory Access (DMA)

- Algoritmo base:
  - Il DMA controller riceve dalla CPU le informazioni necessarie per il trasferimento
  - Il processore prosegue nell'esecuzione del programma in corso
  - Nel frattempo il DMA controller effettua il trasferimento richiesto
  - Al termine del trasferimento, il DMA controller emette un'interruzione per segnalare al processore che l'operazione di I/O è conclusa
- Oss: Il DMA controller deve essere in grado di generare interruzioni [come i moduli di I/O] e di "simulare" la CPU:
  - nel dialogo con la memoria [questione: possibile conflitto d'accesso al bus di memoria]
  - nel dialogo con il modulo di I/O



## □ Direct Memory Access (DMA)

- Modalità di trasferimento DMA:
  - **Block transfer [Bus master]**
    - il DMA controller diventa MASTER del bus per il trasferimento dei dati
    - il processore ne riprende il controllo solo a trasferimento avvenuto
    - modalità più veloce; utile per unità I/O del tipo disco
  - **Transparent**
    - il DMA controller effettua un trasferimento solo quando il bus non è utilizzato dal processore
    - modalità più lenta
  - **Bus stealing [furto di ciclo]**
    - il trasferimento avviene in determinati punti del ciclo di esecuzione delle istruzioni
    - il processore viene interrotto prima che acceda al bus, del quale il DMA assume il controllo per un ciclo

## ❑ Direct Memory Access (DMA)

- Esempio: System bus – I/O bus

