L'input/output

Architettura
degli Elaboratori e delle Reti

Alberto Borghese
Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
email: borghese@dsi.unimi.it

I/O

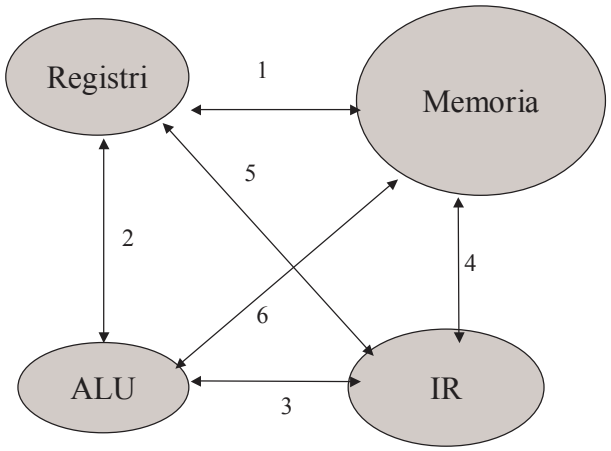
Dispositivi eterogenei per:

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

Dispositivi di I/O - esempi

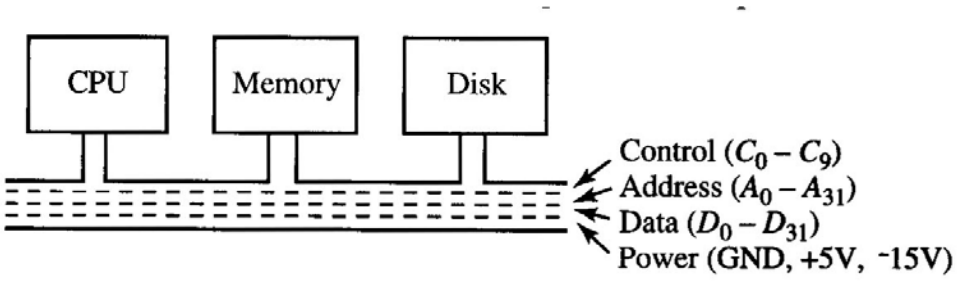
Dispositivo	Comportamento	Partner	Tasso dati (KB/sec)
Tastiera	Input	Umano	0.01
Mouse	Input	Umano	0.02
Stampante laser	Output	Umano	100.00
Floppy disk	Memoria	Macchina	50.00
Disco ottico	Memoria	Macchina	500.00
Disco magnetico	Memoria	Macchina	10,000.00
Rete-LAN	Input o Output	Macchina	20 – 1,000.00
Video grafico (AGP)	Output	Umano	100,000.00

Collegamento tra le unità



Connessione di tutti i dispositivi con tutti.
- N * (N-1) connessioni bidirezionali (difficili da controllare).
- Non c'è questa necessità.

Il bus (connessione a nodo comune)



Pathway che connette tutti i dispositivi in modo bidirezionale.

Principali vantaggi della struttura a bus singolo:
• elevata flessibilità
• bassi costi.

Bus – il contenuto

- Permette la comunicazione tra le diverse unità del calcolatore ed è generalmente composto da tre parti logiche:
  - ***Bus dati***, comprende le linee per trasferire dati e istruzioni da/verso I dispositivi (la memoria). In generale, la dimensione del bus dati è tale da garantire il trasferimento contemporaneo di una o più parole 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 (esempio la cella di memoria).
  - ***Bus di controllo***, dove transitano le informazioni ausiliarie per la corretta definizione delle operazioni da compiere (per esempio l'indicazione che si vuole effettuare una *lettura* piuttosto che una *scrittura*) e per la sincronizzazione tra *CPU* e memoria.
- **Protocollo** La comunicazione su bus deve essere regolata attraverso un **protocollo di comunicazione**.

Bus di sistema & buffer

- 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.

Arbitraggio del bus

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.

Questo è il principale inconveniente dei bus a nodo comune.

Tipologie di bus

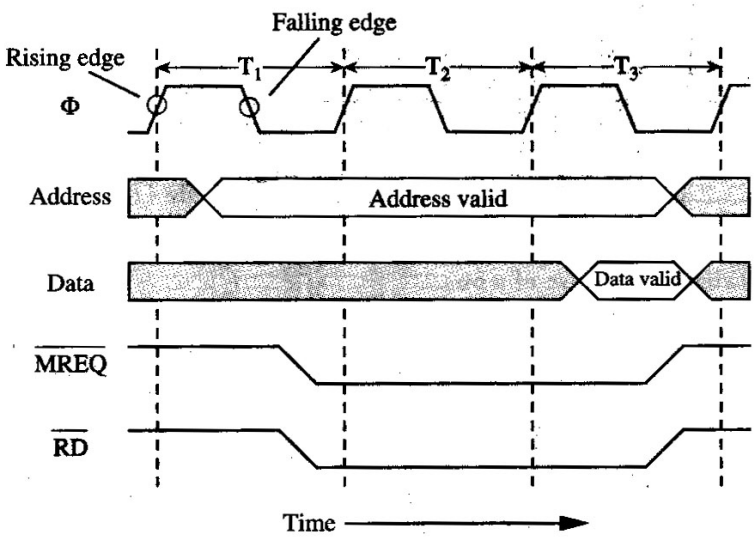
- Tre tipi principali:
  - **processor-memory (cache)**: lunghezza ridotta, molto veloci.
  - **I/O**: notevole lunghezza, molti device connessi.
  - **backplane:** servono per far coesistere la memoria, il processore e i dispositivi di I/O su di un unico bus.
- Esistono due schemi principali di comunicazione su di un bus:
  - Sincrono
  - Asincrono

Bus sincroni

- La linea di controllo è dotata di un segnale di sincronizzazione **(bus clock)** ed 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).
  - 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

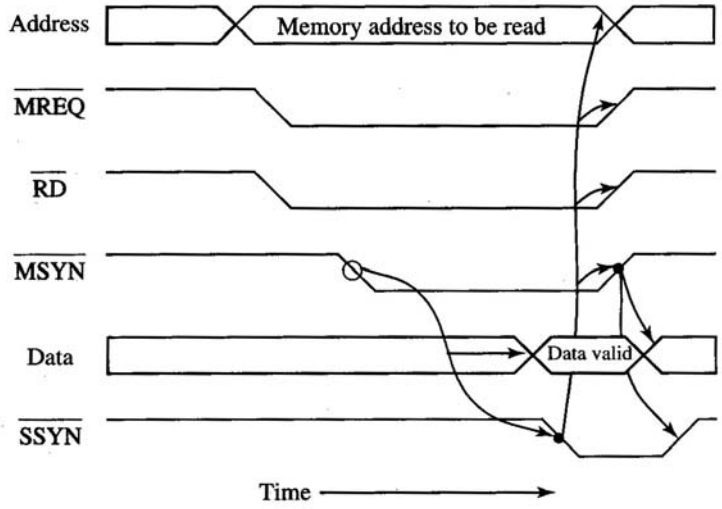


Bus asincroni

- 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 e connettere molti dispositivi.
- Spesso i bus di I/O sono asincroni.

Bus asincroni: esempio



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*)

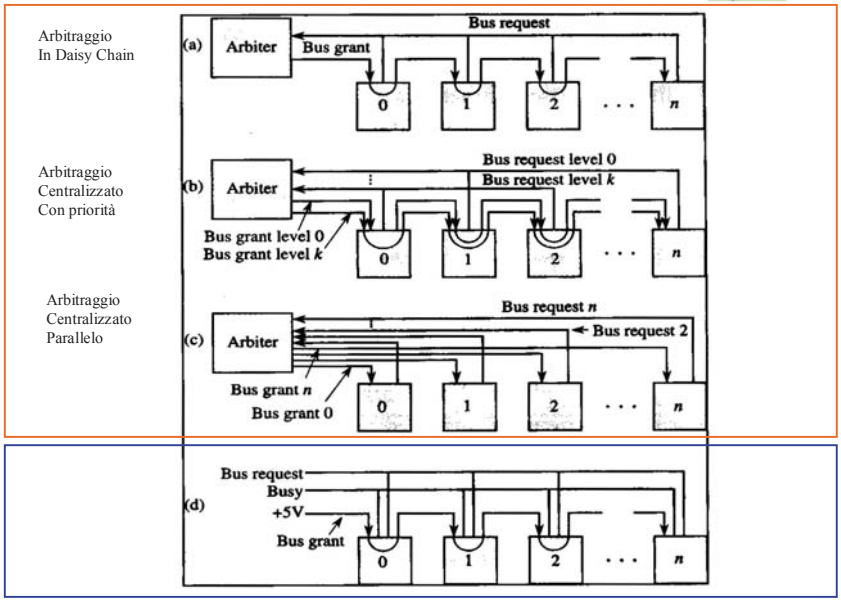
Problema è assicurare una *fairness*.

Compromesso tra fairness e priorità.

Schemi di arbitraggio

Schemi centralizzati

Schema Decentralizzato (distribuito)

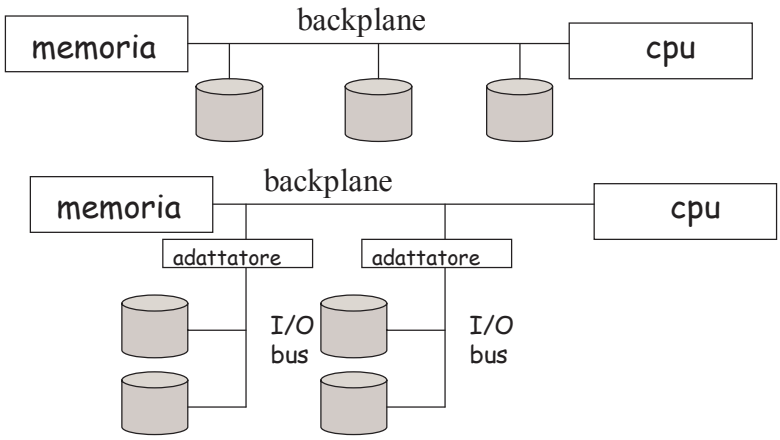


Arbitraggio distribuito con autoselezione

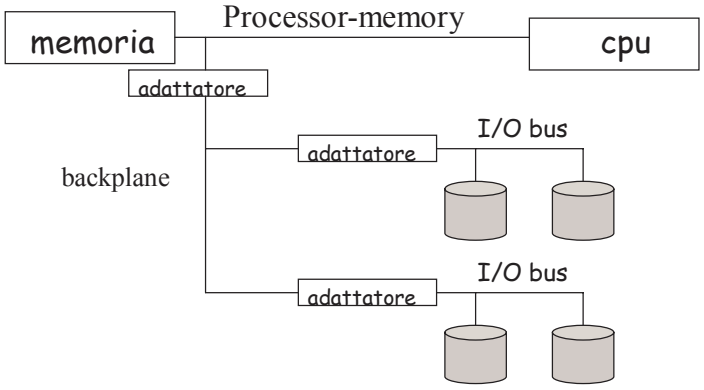
In questo schema un dispositivo che vuole prendere il controllo del bus, deve:

- 1) Inviare il segnale di richiesta del bus.
- 2) Scrivere sul bus il codice che lo identifica.
- 3) Controllo 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.
- 6) Deasserire la richiesta del bus.

Tipologie di bus



Il bus CPU-Memoria



Gli adattatori sono dispositivi attivi – bridges.

Esempio – Pentium II

Bus processore-memoria (cache)

Bus di sistema (processore-RAM)

Bus di backplane

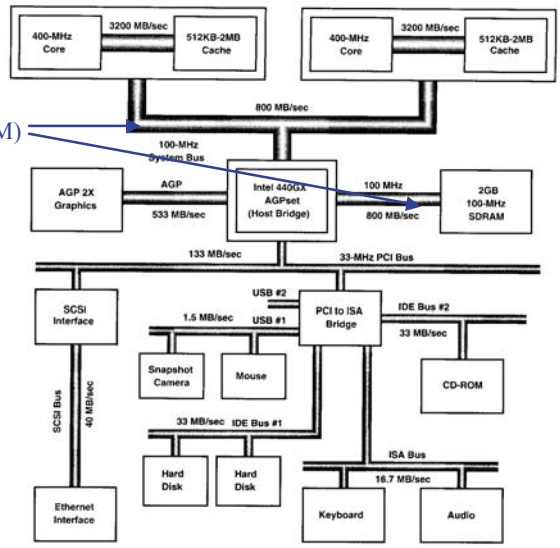
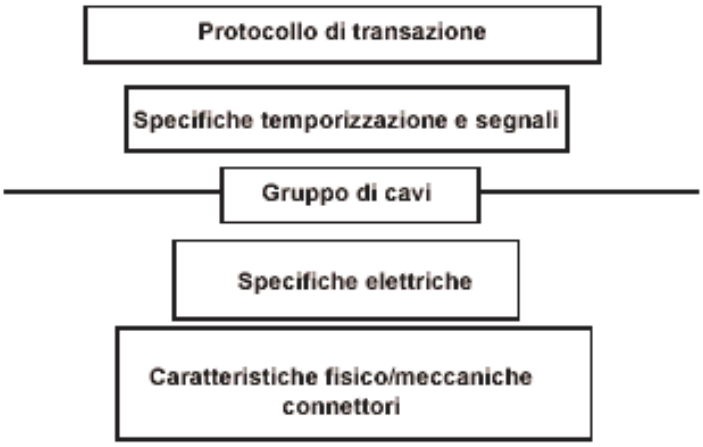


Figure 8-7 Bridging with dual Pentium II Xeon processors on Slot 2.
(Source: http://www.intel.com)

I bus

Caratteristica	PCI	Firewire (IEEE 1394)	SCSI
Tipo di bus	Back-plane	Backplane & I/O (peer-to-peer)	I/O
Ampiezza di base del bus (numero di segnali per i 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)
Ampiezza di banda di picco teorica	133-512MB/s (PCI64)	266MB/s (IEEE 1394b)	5-40MB/s
Ampiezza di banda stimata raggiungibile per bus di base	80MB/s	10-100MB/s	2,5-40MB/s
Massimo numero di dispositivi	1024 (32 dispositivi per segmento)	63	7-31
Massima lunghezza del bus	0,5 metri	0,5 metri (sincrono)	25 metri
Nome dello standard	PCI	IEEE 1394	ANSI X3.131

Caratteristiche di un bus



Il bus: Riassunto

- Esempio: operazione di lettura dalla memoria. La *CPU* fornisce l'indirizzo della parola desiderata sul bus indirizzi, quindi viene richiesta l'operazione di lettura attivando il bus di controllo. Quando la memoria ha completato la lettura della parola richiesta, il dato viene trasferito sul bus dati e la *CPU* può prelevarlo ed utilizzarlo nelle sue elaborazioni.
- La struttura del bus può essere realizzata secondo diverse topologie di interconnessione.
- Il bus può esser utilizzato per un solo trasferimento alla volta ⇒ in ogni istante soltanto due unità *(Master e Slave)* possono usare il bus.
- Nelle architetture di riferimento l'unità Master è solitamente la CPU. Esistono alcune schede di I/O particolarmente performanti che possono diventare anch'esse bus-master.
- Le linee di controllo del bus vengono utilizzate per inviare più richieste contemporanee di utilizzo del bus che vengono gestite dalla logica di *arbitraggio* del bus.

Aumento della banda passante del bus

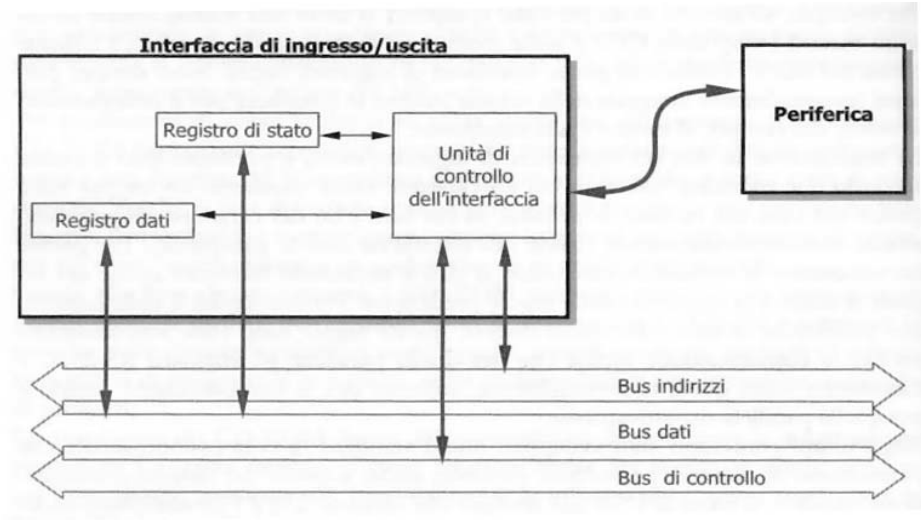
• **Linee dati ed indirizzo separate:**
– **indirizzi e dati possono viaggiare nello stesso ciclo di bus**
– **costo: (a) più linee di bus, (b) maggiore complessità**

• **Larghezza del bus dati:**
– **aumentando la larghezza del bus dati, trasferimento di più parole richiede meno cicli di bus**
– **esempio: SPARCstation con 20 bus di memoria larghi 128 bit**
– **costo: più linee di bus**

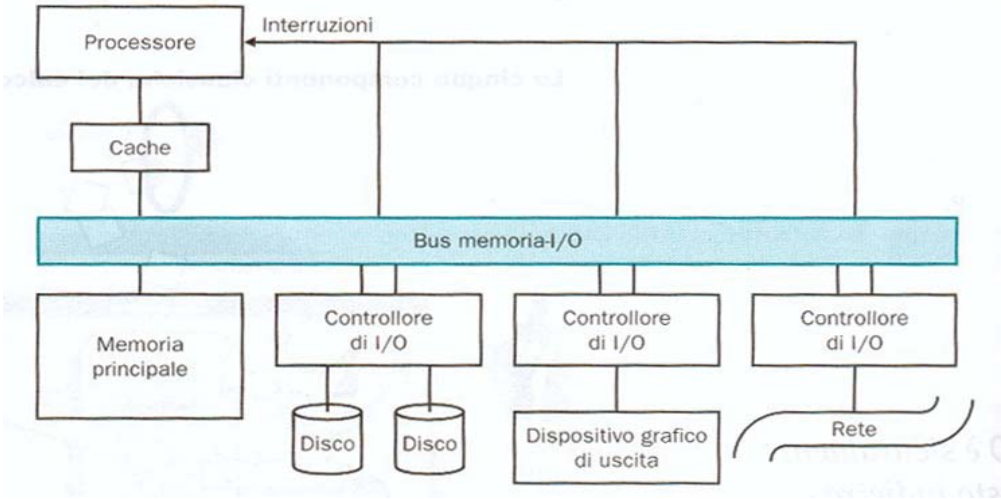
• **Trasferimento di blocchi:**
– **permettere trasferimento di più parole in cicli di bus adiacenti**
– **solo un indirizzo deve essere spedito all'inizio**
– **il bus non viene rilasciato fino a che l'ultima parola è stata trasferita**
– **costo: (a) maggiore complessità; (b) diminuzione del tempo di risposta per richiesta**

Gestione dell'I/O

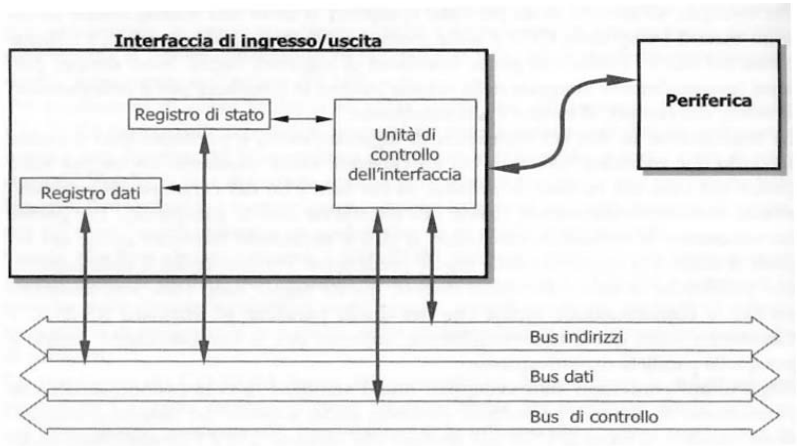
Il device controller



I driver (o controllori)

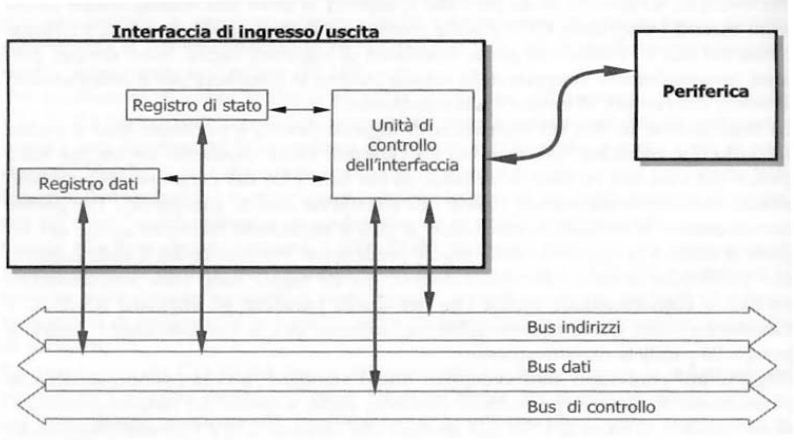


Modalità trasferimento dati



Parallela (centronics). 1 byte alla volta.
Seriale (RS232, RS432). 1 bit alla volta. Nuovi standard sono USB e Firewire (IEEE 1394) e Bluetooth (wireless).

Elementi di interfaccia



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

Indirizzamento dell'I/O

- Memory-mapped
- Istruzioni speciali di I/O

Istruzioni speciali di I/O

Istruzioni appartenente alla ISA che indirizzano direttamente il dispositivo:

- Numero 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 l'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.

• 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.

Gestione dell'I/O

- A controllo di programma diretto
- A controllo di programma con polling
- Ad interruzione
- Ad accesso diretto alla memoria (DMA)

I/O a controllo di programma

e.g. chiamata syscall per la stampa di una stringa.

La periferica ha un ruolo passivo. Il processore esegue tutto il lavoro.

Svantaggio: La CPU dopo avere predisposto il controller all'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. Stato *busy waiting* o *spin lock*.

```
begin
        1. Predisponi i registri del controller ad effettuare una
                operazione di lettura.
        2. While (ready-bit = = 0) do;          // spin lock
        3. Carica il dato acquisito;
end;
```


Esempio: Receiver

```
.text
.globl main
main:
        li $t0, 0xffff0000      # indirizzo del receiver control register
        li $t2, 0xffff0004      # indirizzo del receiver data register

# Ciclo di lettura di un carattere
ciclo:  lw $t1, 0($t0)          # Contenuto del registro di controllo
        rem $t1, $t1, 2         # If $t1 = = 1 (resto = 1) esci
        beqz $t1, ciclo
        lw $a0, 0($t2)

        li $v0, 10
        syscall
```

I/O a gestione di programma - costo

Ipotesi:
1) Tastiera gestita a controllo di programma che opera a 0,01Kbyte/s.
2) Frequenza di clock: 50Mhz.

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 = 4 / 0,01 Kbyte /s = 0,4s #cicli_clock = $50*10^6 * 0,4 = 20,0 * 10^6$

Invece ne utilizza solo 20 * 4 = 80 per trasferire i dati.

%sfruttamento della CPU è: $(80/20,0 * 10^6) *100 = 0,0004\%$

I/O a gestione di programma - costo

Ipotesi:
1) Floppy che opera a 50Kbyte/s.
2) Hard-disk che lavora a 2MByte /s.
3) Frequenza di clock: 50Mhz.

Determinare la percentuale di tempo in cui verrebbe effettivamente utilizzata la CPU per trasferire 1 parola, tenendo conto che ci vogliono 20 cicli di clock per ogni byte (approssimare 1k = 1,000).

Floppy-disk: 4 byte / 50kbyte/s = 80µs =>
$50x10^6$/s (#cicli_clock/tempo) * $80x10^{-6}$s (tempo_trasferimento) = 4000#cicli_clock => %sfruttamento = $(20*4) / 4x10^3 = 2\%$

Hard-disk: 4 byte / 2Mbyte /s = 2µs =>
$50x10^6$/s * $2x10^{-6}$ = 100 cicli_clock
=> %sfruttamento = (20*4) / 100 = 80%