Università degli Studi di Cassino

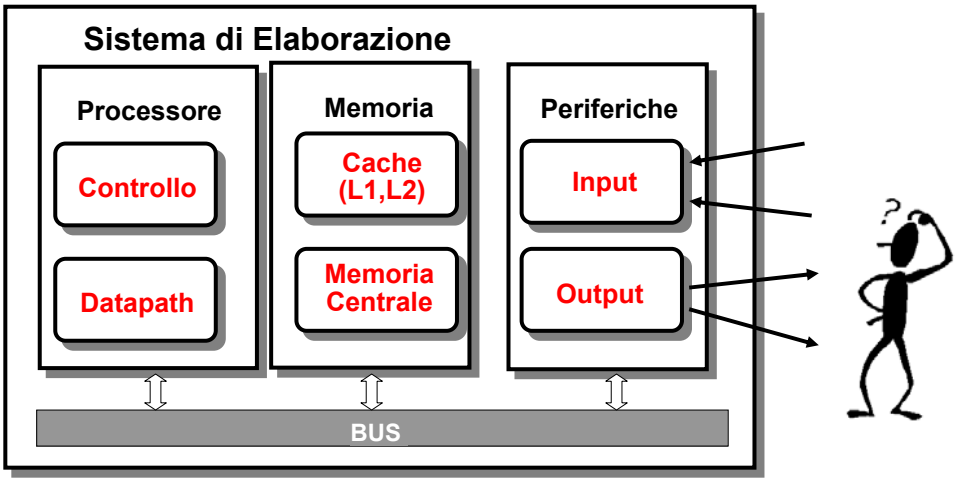
Corso di Calcolatori Elettronici II

Input/Output

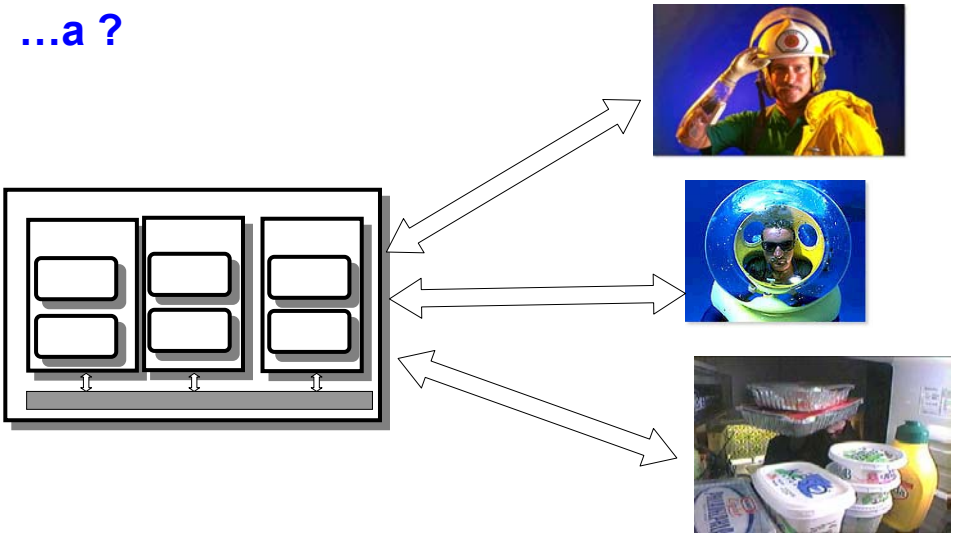
Anno Accademico 2004/2005

Francesco Tortorella

Input/Output: da un sistema di elaborazione a … ?



…a ?



Motivazioni per l' Input/Output

- Il sistema di I/O definisce l'interazione tra calcolatori ed esseri umani e permette operazioni notevoli:

– Acquisisce dati sensoriali (pressione, temperatura) dalla mano sintetica del vigile del fuoco Ken Whitten e li inoltra sul sistema nervoso.

– Controlla la propulsione, l'assetto, gestisce la comunicazione in BOB (Breathable Observable Bubble)

– Legge I codici a barre degli articoli nel frigorifero (e prepara la lista della spesa)

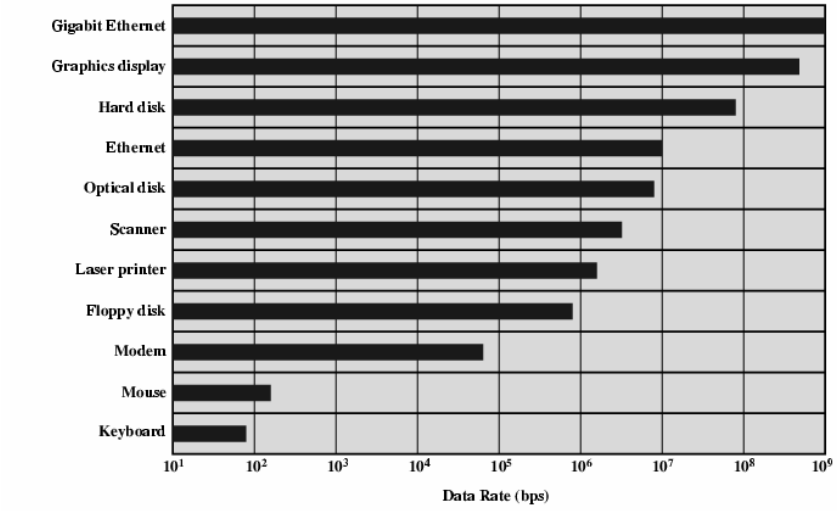
Computer without I/O like a car without wheels; great technology, but won't get you anywhere

Periferiche “tradizionali”

Periferica	Funzione	Partner	Velocità (Kbytes/sec)
Tastiera	Input	umano	0.01
MouseInput	umano	0.02	
Stampante	Output	umano	1.00
Floppy disk	Memorizz.	macchina	50.00
Stampante Laser	Output	umano	100.00
Disco Ottico	Memorizz.	macchina	500.00
Disco Magnetico	Memorizz.	macchina	10000.00
Network-LAN	I/O	macchina	10000.00
Display Grafico	Output	umano	30000.00

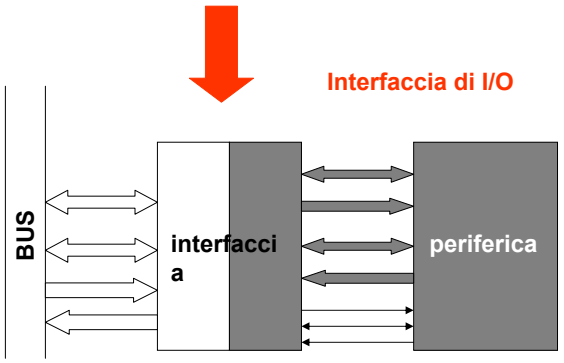
grossa variabilità tra le periferiche

Periferiche “tradizionali”: la classifica

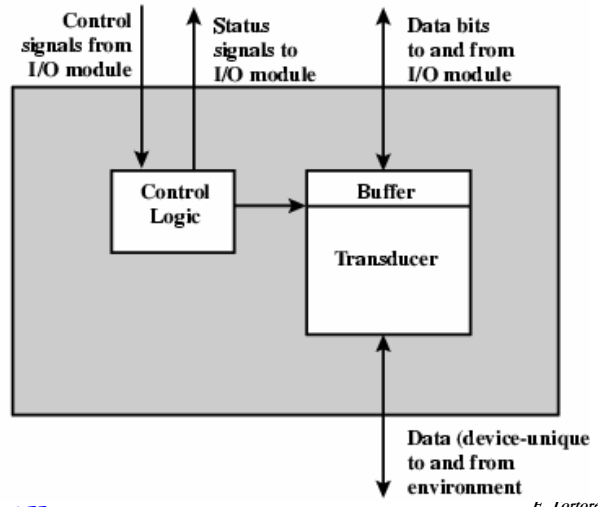


Interfaccia di I/O

- Molti tipi diversi di periferiche
- Grossa variabilità all'interno di una stessa classe di periferiche
- Necessità di uniformare la connessione tra la periferica ed il resto del sistema



Interfaccia di I/O: schema della periferica



Funzioni dell'interfaccia di I/O

- Controllo e tempificazione
- Comunicazione con la CPU
- Comunicazione con la periferica
- Buffering dei dati
- Error Detection

Fasi di un'operazione di I/O

- La CPU verifica lo stato dell'interfaccia di I/O
- L'interfaccia di I/O fornisce lo stato
- Se lo stato è ready, la CPU richiede un trasferimento di dati
- L'interfaccia di I/O acquisisce i dati dalla periferica
- L'interfaccia di I/O trasferisce i dati alla CPU

- **ACHTUNG**: descrizione generica – possibili molte variazioni sul tema !!

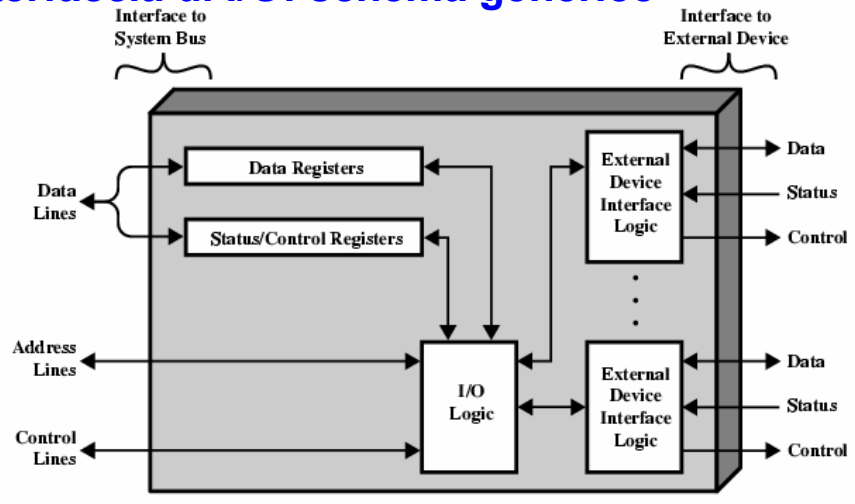
Interfaccia di I/O

- L'interfaccia è accessibile tramite bus in maniera analoga ad un modulo di memoria (viene selezionata dal processore tramite un indirizzo)
- Le comunicazioni avvengono tramite operazioni di lettura/scrittura su un insieme di registri (*I/O ports*) appartenenti all'interfaccia:
  - Data IN: registro dati in ingresso
  - Data OUT: registro dati in uscita
  - Control Register (in): riceve istruzioni per le operazioni della periferica
  - Status Register (out): contiene informazioni sullo stato della periferica
- L'accesso ai registri è multiplexato:
  - interfaccia mappata su due indirizzi, multiplexati in funzione di un segnale di controllo READ/WRITE

    IND1: Data IN / Data OUT

    IND2: Control / Status

Interfaccia di I/O: schema generico



I/O Addressing

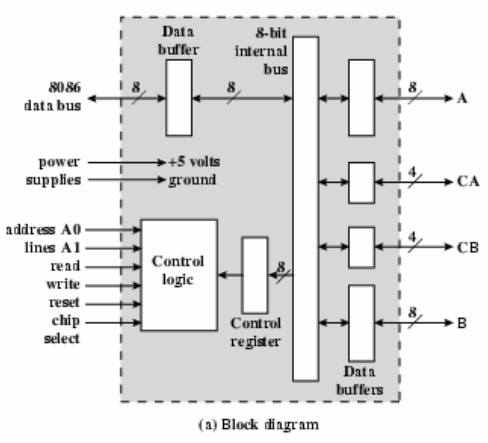
Ogni interfaccia viene selezionata dal processore tramite un indirizzo univoco.

Questo può appartenere:

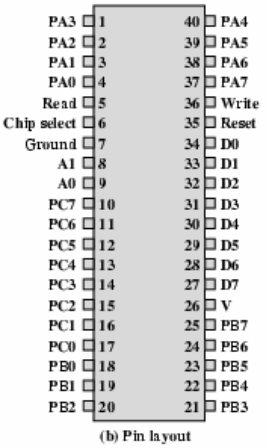
- allo spazio di indirizzamento della CPU ⇨ **Memory mapped I/O**
  - operazioni di I/O realizzate tramite istruzioni di MOVE
  - numero di I/O ports teoricamente illimitato
  - struttura del bus più semplice
- ad uno spazio di indirizzamento separato ⇨ **Independent I/O**
  - presenza di istruzioni specifiche per l'I/O (IN, OUT)
  - non si sacrificano indirizzi di memoria per l'I/O
  - struttura del bus più complessa

Possibili soluzioni miste

Intel 82C55A
Programmable Peripheral Interface

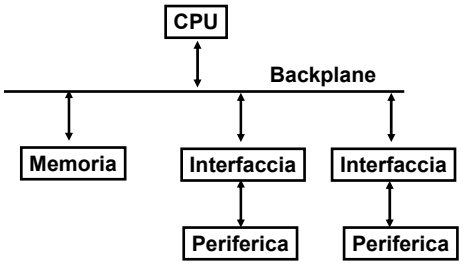


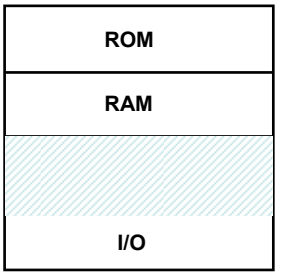
(a) Block diagram

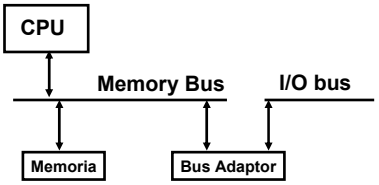


(b) Pin layout

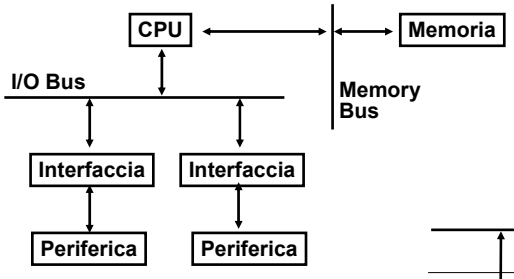
Memory mapped I/O

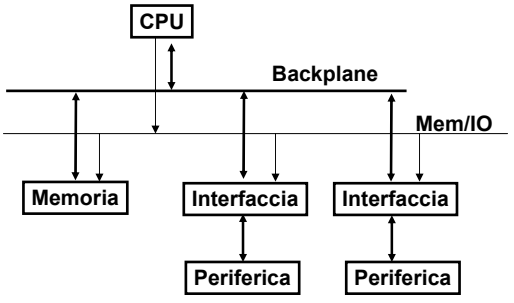






Independent I/O



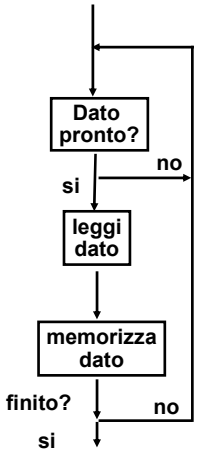


Tempificazione delle operazioni di I/O

- Enorme differenza tra la velocità della CPU e delle periferiche
- Un processore funzionante a 500 MHz può (teoricamente) gestire un traffico di dati ad una velocità di 200000 KB/s, mentre le periferiche più veloci arrivano a 30000 KB/s
- **Input**: la periferica può non essere in grado di inviare dati alla CPU alla stessa velocità con cui il processore riesce a leggerli. Inoltre, la periferica potrebbe interagire con un utente umano, rallentando ulteriormente l'attività
- **Output**: la periferica può non essere in grado di accettare dati dalla CPU alla stessa velocità con cui il processore li produce
- Non è realizzabile una gestione sincrona delle operazioni di I/O.

Gestione delle operazioni di I/O: Polling

- Gestione asincrona dell'operazione
- La CPU inizia, dirige e termina l'operazione di I/O, rimanendo in attesa del completamento (*Programmed I/O*)
- Esempio: Realizzazione della lettura di un dato
  - Inizio: la CPU accede al Control Register dell'interfaccia, scrivendo una word che inizializza l'interfaccia e la predispone per l'operazione richiesta
  - Attesa: la CPU accede in lettura allo Status Register dell'interfaccia, rimanendo in loop finchè il dato non è disponibile (*busy waiting*)
  - Terminazione: la CPU accede in lettura al Data OUT, prelevando il dato ed inizializzando lo Status Register



E' conveniente il polling ?

- Con la tecnica del polling, la CPU spende del tempo in attesa della periferica:
  - quanto tempo viene perso in ogni interrogazione ?
  - con che frequenza avvengono le interrogazioni ?
- Consideriamo un processore con un clock da 500 MHz; per ogni interrogazione vengono impiegati 400 cicli di clock (chiamata alla procedura di polling, accesso all'interfaccia, ritorno). Valutiamo il carico dovuto al polling per periferiche diverse
  - Mouse: interrogato 30 volte al secondo per non perdere movimenti dell'utente
  - Floppy disk: trasferimento dati in unità da 2 byte, con una velocità di trasferimento di 50 KB/sec. E' necessario non perdere nessun dato.
  - Hard disk: trasferimento dati in blocchi da 16 byte, con una velocità di trasferimento di 8 MB/sec. Anche qui è necessario non perdere alcun dato.

Un po' di conti … (1/2)

- Mouse:
  - cicli al secondo necessari per l'interrogazione:
    30 * 400 = 12000 cicli/sec
  - overhead sul processore:
    12*103/500*106 = 0.002%
- ⇒ Polling con un impatto trascurabile sul processore
- Floppy:
  - interrogazioni/sec = 50 KB/s /2B = 25K poll/sec
  - cicli al secondo necessari per l'interrogazione:
  - 25K * 400 = 107 cicli/sec
  - overhead sul processore :
  - 10*106/500*106 = 2%
- ⇒ ancora accettabile se non ci sono molte periferiche

Un po' di conti … (2/2)

- Hard Disk:
  - interrogazioni/sec = 8 MB/s /16B = 500K poll/sec
  - cicli al secondo necessari per l'interrogazione:
  - 500K * 400 = 2*108 cicli/sec
  - overhead sul processore :
  - 200*106/500*106 = 40%
- $\Rightarrow$ inaccettabile

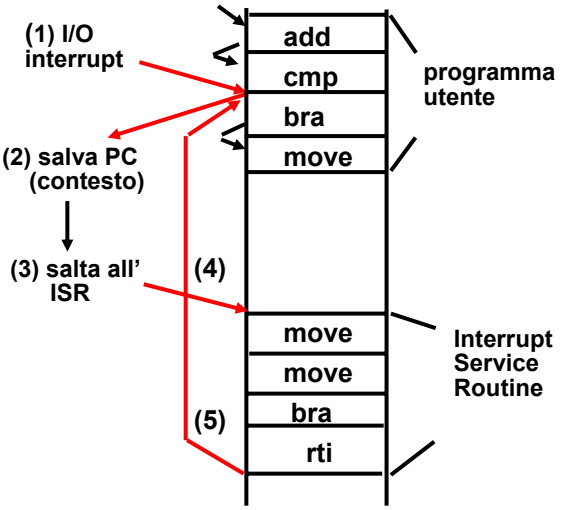
Alternative al polling - Interrupt I/O

- E' inutile che il processore sprechi tempo nell'attesa che la periferica sia pronta.
- Sarebbe sufficiente che la procedura di lettura fosse eseguita solo quando la periferica è pronta.

Uso delle interruzioni
- Possibile una gestione asincrona dell'operazione di I/O
- L'interfaccia deve essere in grado di generare un segnale di interrupt
- La CPU non deve attendere il completamento dell'operazione, ma risponde alla richiesta di interruzione

Gestione di un'operazione via interrupt I/O

- La CPU inizia l'operazione, scrivendo il contenuto del C.R. dell'interfaccia.
- Non rimane in attesa del completamento, ma riprende il programma, interrompendolo su richiesta dell'interfaccia



Valutazione dell'Interrupt I/O

- Supponiamo un overhead di 500 cicli di clock per ogni trasferimento, interrupt inclusa. Supponiamo che l'hard disk sia attivo il 5% del tempo.
- Interrupt rate
  - Disk Int/sec = 8 MB/s /16B
    = 500K int/sec
  - Disk Polling cicli/sec = 500K * 500
    = 250,000,000 cicli/sec
  - Overhead sul processore durante i trasferimenti:
    250*106/500*106= 50%
- Disco attivo 5% $\Rightarrow$ 5% * 50% = 2.5% overhead totale

Interrupt I/O: questioni aperte

- Quale device ha generato l'interruzione?
  - Necessario affiancare l'informazione sull'identità del device
- E' possibile evitare interruzioni durante la gestione dell'interrupt ?
  - Che cosa succede se arrivano interrupt a maggiore priorità ?
  - E' possibile rientrare su una "interruzione interrotta" ?

Soluzioni possibili

- Identificazione del dispositivo interrompente
  - Multiline
  - Daisy chain
  - Polling
  - Interrupt vettorizzato
- Gestione della priorità
  - Gestione interna al processore
  - Gestione esterna al processore
- Programmable Interrupt Controller

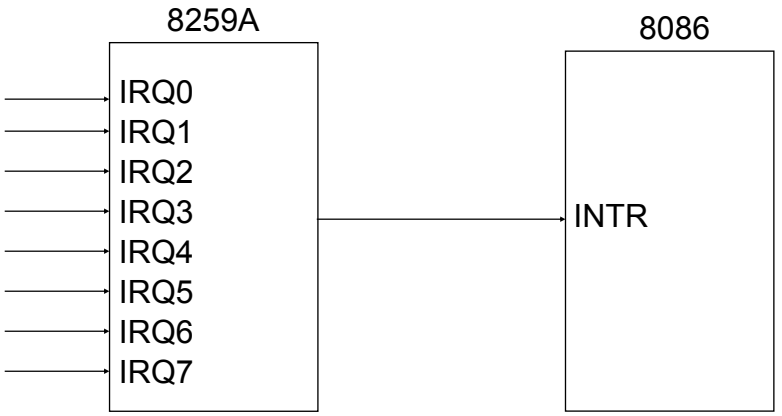
Esempio: l'80x86

- L'80x86 ha una sola linea d'interrupt
- Viene utilizzato il PIC 8259A
- L'8259A ha 8 linee d'interrupt
- Può essere montato in cascata, aumentando così le linee di interrupt disponibili

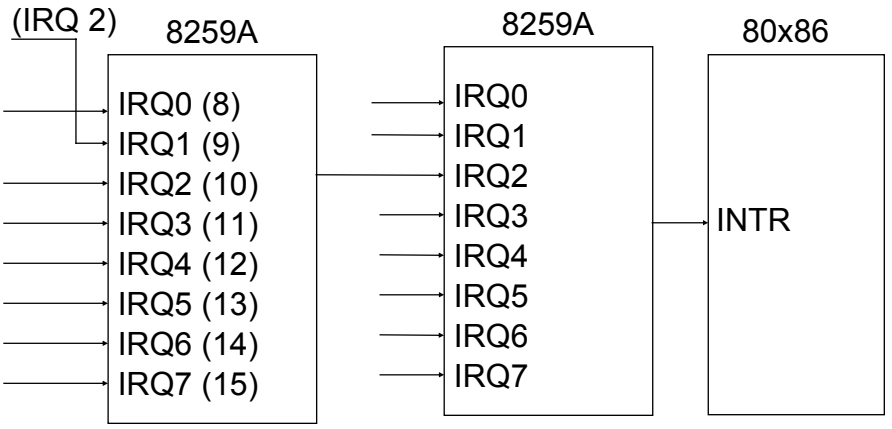
Esempio di PIC: l'8259

- Sequenza di eventi:
  - 8259A riceve gli interrupt
  - 8259A determina la priorità
  - 8259A segnala l'interrupt al processore 8086 (attiva la linea INTR)
  - La CPU accetta l'interrupt (ACK)
  - 8259A fornisce il vettore opportuno sul data bus
  - La CPU serve l'interrupt

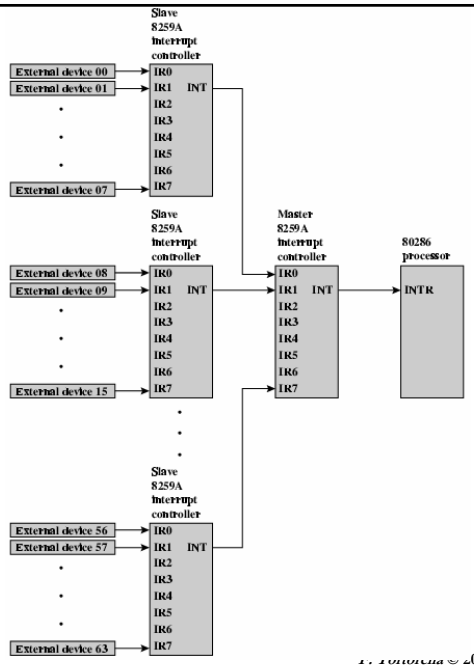
Esempio di PIC: l'8259



Esempio di PIC: l'8259



Esempio di PIC: l'8259



Gestione dell' I/O: questioni aperte

- L'organizzazione e la gestione delle operazioni di I/O ha una complessità molto elevata che non può essere gestita dall'utente normale, il quale
  - Non ha conoscenza del sistema di I/O
  - Non ha le competenze necessarie
- E' necessario che questa gestione avvenga a livello di sistema
- Chi mantiene traccia dello stato di tutti i device, gestisce gli errori, sa come è organizzato il sistema di I/O ?

Il ruolo del Sistema Operativo

- **Il sistema di I/O è condiviso tra molti processi che si contendono l'uso del processore**
  - **possibili violazioni di spazi riservati**
  - **possibile sbilanciamento tra gli accessi alle risorse di I/O**
- **Il controllo delle operazioni di I/O è complesso**
- **Le periferiche impiegano spesso le interruzioni per comunicare i risultati delle operazioni di I/O**

Il Sistema Operativo

→ **garantisce che il programma utente accede solo quella parte del sistema di I/O su cui ha diritto**

→ **stabilisce una politica di accesso alle risorse di I/O che assicuri la distribuzione equa degli accessi e l'efficienza complessiva del sistema**

→ **virtualizza l'accesso alle periferiche fornendo procedure che gestiscono le operazioni a basso livello sul device (drivers)**

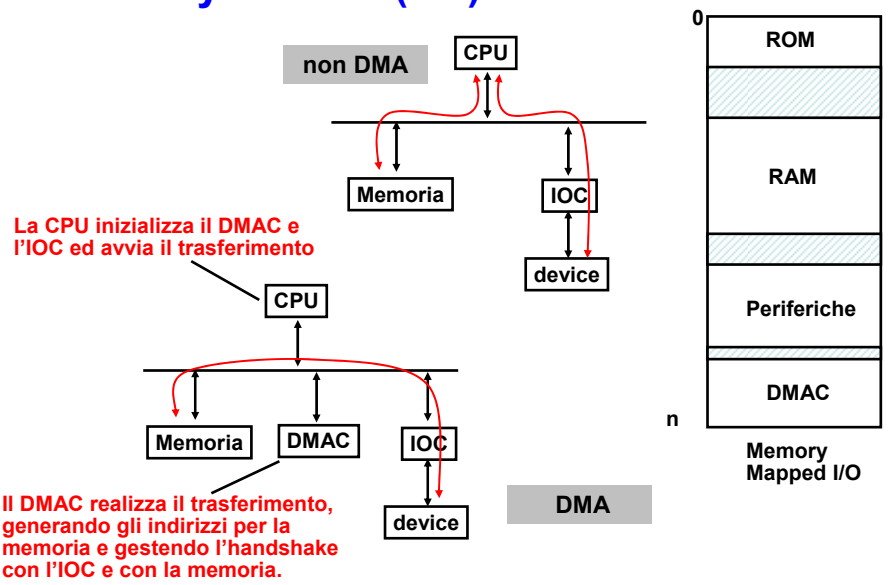
→ **gestisce le interruzioni generate dalle periferiche**

Direct Memory Access (1/3)

- Nel trasferimento di blocchi di dati l'interrupt I/O può risultare comunque pesante
- Durante il trasferimento di un blocco di dati, la CPU è occupata nell'inoltro di dati tra memoria e periferica
- Possibile liberare il processore dalla realizzazione del trasferimento, lasciandogli solo la supervisione dell'operazione (specificare il blocco da trasferire, gestire l'avvio e la chiusura) ?

Direct Memory Access (DMA):
l'operazione è realizzata tramite un dispositivo (DMA controller) capace di trasferire un blocco di dati tra la memoria ed una periferica e di generare un interrupt al termine

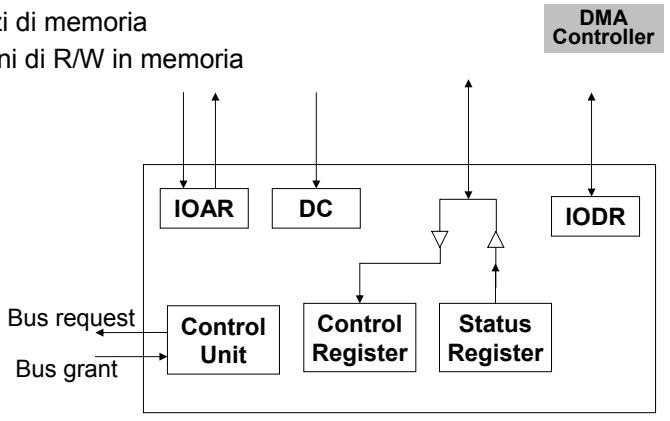
Direct Memory Access (2/3)



Direct Memory Access (3/3)

Per realizzare l'accesso diretto in memoria, il DMA controller deve essere capace di:

- gestire la richiesta per l'uso del BUS
- generare gli indirizzi di memoria
- realizzare operazioni di R/W in memoria



Registri del DMAC:

IOAR **(I/O Addr. Reg.)**
DC **(Data Count)**
IODR **(I/O Data Reg.)**

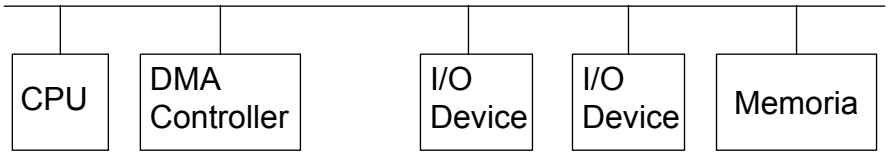
Sequenza di operazioni di un trasferimento DMA

- 1. La CPU fornisce le informazioni relative al blocco da trasferire ed inizializza il DMAC:
  - 1.1 base address → IOAR
  - 1.2 data count → DC
  - 1.3 controllo → Control Register
- 2. DMAC asserisce BUS REQUEST
- 3. La CPU rilascia il bus e asserisce BUS GRANT
- 4. Acquisito il bus, il DMAC trasferisce i dati da/verso la memoria incrementando IOAR e decrementando DC
- 5. Se DC $\neq$ 0 e il device esterno non è pronto, il DMAC rilascia il bus che ritorna alla CPU
- 6. Se DC = 0, il controllo ritorna alla CPU. Eventualmente, viene generata un'interruzione

DMA - Modalità di trasferimento

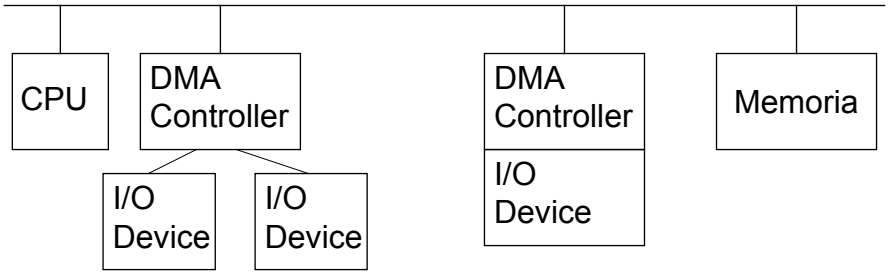
- **Trasferimento a blocco** (block transfer)
  - L'intero blocco viene trasferito mentre il DMA Controller è master del bus
  - Necessario per periferiche (p. e. dischi) in cui il trasferimento non può essere interrotto o rallentato
- **Furto di ciclo** (cycle stealing)
  - Il blocco viene trasferito attraverso una sequenza di trasferimenti di blocchi di lunghezza predefinita, alternandosi con la CPU sul bus
- **Trasferimento trasparente** (trasparent DMA)
  - Il trasferimento ha luogo solo quando la CPU non è master del bus

Configurazione DMA (1)



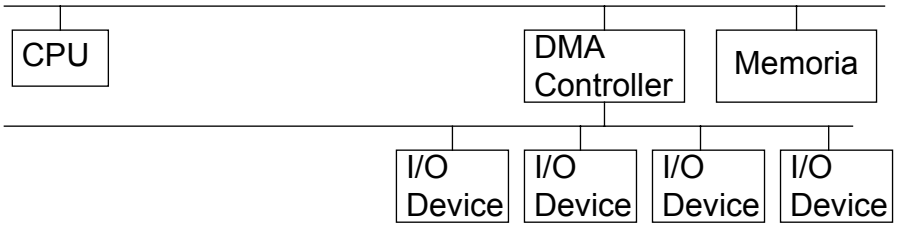
- Bus singolo, DMA controller separato
- Ogni trasferimento richiede due transazioni di bus
  - I/O → DMA quindi DMA → memory
- La CPU è sospesa due volte

Configurazione DMA (2)



- Bus singolo, DMA controller integrato
- Ogni controller può essere collegato a più di un device
- Ogni trasferimento richiede una transazione di bus
  - DMA → memory

Configurazione DMA (3)



- Bus di I/O separato
- Il bus supporta tutti i device gestiti via DMA
- Ogni trasferimento richiede una transazione di bus
  - DMA → memory

Valutazione dell'overhead relativo al DMA

- Frequenza processore: 500 MHz
- Setup del DMAC: 1000 cicli
- Servizio dell'interruzione: 500 cicli
- Hard Disk connesso via DMA
- Velocità di trasferimento: 8 MB/s
- Dimensione media del blocco trasferito: 8 KB

Tempo per trasferimento di un blocco: 8 KB / 8 MB/s = 10^{-3} sec

Disco sempre attivo ⇒ 10^3 trasf/sec

Overhead del processore ⇒ (1000+500) * 10^3 cicli/sec

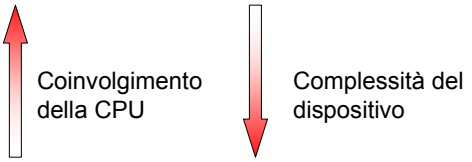
% Overhead ⇒ $1.5*10^6 / 500*10^6 = 0.3$ %

I processori di I/O

programmed I/O

interrupt I/O

DMA



Dispositivi di I/O dalle più ampie capacità liberano la CPU dal peso della gestione delle operazioni di I/O.

Il costo da pagare è la complessità del dispositivo di I/O.

Soluzione estrema: **un processore dedicato all'I/O**

I processori di I/O

I processori di I/O (o *canali*) sono vere e proprie CPU dedicate esclusivamente alla gestione delle operazioni di I/O.

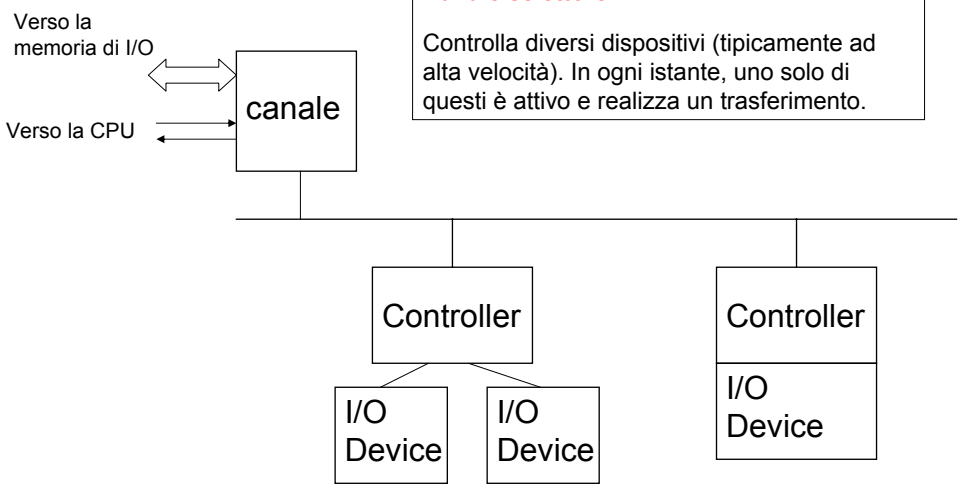
Il codice da eseguire (programma di I/O) è contenuto in una memoria riservata e realizza tutte le fasi delle operazioni di I/O (inizializzazione, trasferimento, chiusura, gestione di errori).

La CPU avvia l'operazione di I/O semplicemente fornendo al processore di I/O l'indirizzo del programma di I/O da eseguire.

Al termine dell'operazione, il processore di I/O produce un'interruzione per avvertire la CPU.

In questo modo, la CPU non ha più diretto controllo sul sistema di I/O.

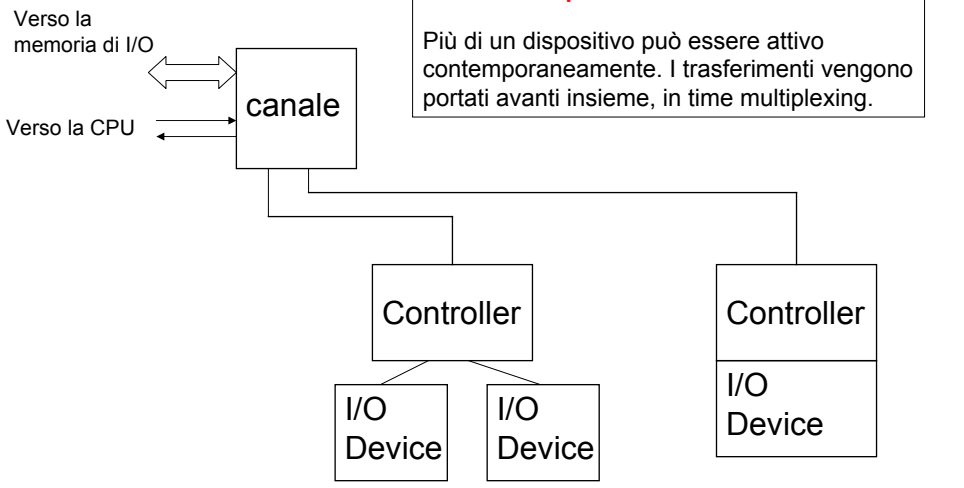
Configurazione dei canali (1)



Canale selettore

Controlla diversi dispositivi (tipicamente ad alta velocità). In ogni istante, uno solo di questi è attivo e realizza un trasferimento.

Configurazione dei canali (2)



Canale multiplexer

Più di un dispositivo può essere attivo contemporaneamente. I trasferimenti vengono portati avanti insieme, in time multiplexing.