

# Corso di Calcolatori Elettronici

## Il sistema di Input-Output

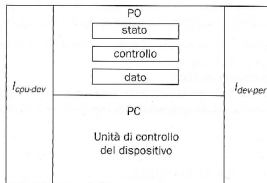
Roberto Canonico

Università degli Studi di Napoli Federico II

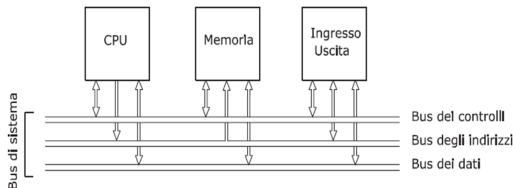
A.A. 2022-2023



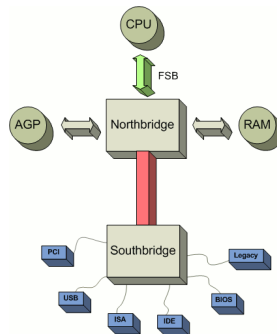
- L'interazione tra la CPU ed una generica periferica di I/O avviene tramite una *interfaccia*
- L'interfaccia si presenta alla CPU come un insieme di registri che ne costituiscono il *modello di programmazione*
- I registri mantengono informazioni riconducibili a tre classi fondamentali: dato-stato-controllo
- $I_{CPU-dev}$  collega l'interfaccia ai bus di sistema
- $I_{dev-per}$  collega l'interfaccia alla periferica



- Nei primi calcolatori un unico bus di sistema collegava la CPU sia con la memoria centrale che con i sistemi di I/O



- Nei calcolatori moderni il sistema di interconnessione è organizzato gerarchicamente
- Appositi dispositivi (bridge) realizzano l'accoppiamento tra i vari bus
- Al north bridge (più vicino alla CPU) sono collegati bus ad elevata velocità
  - Collegamenti con la memoria centrale e con il processore grafico GPU (bus AGP)
- Al south bridge si collegano le periferiche tramite bus PCI
- Al south bridge si collegano tramite adattatori anche altri bus più lenti





- I registri della interfaccia di una periferica saranno accessibili dalla CPU mediante appositi indirizzi
- Sono possibili due diversi schemi:
  - I/O isolato
  - I/O memory-mapped
- Nell'*I/O isolato* i registri delle interfacce di I/O sono collocati in uno spazio di indirizzamento autonomo
  - Ciascun registro è associato ad una cosiddetta *porta di I/O*
  - Le operazioni di lettura e scrittura sui registri di interfaccia si realizzano mediante istruzioni I/m dedicate (es. IN e OUT)
- Nell'*I/O memory-mapped* i registri delle interfacce di I/O sono collocati nello spazio di indirizzamento della memoria centrale
  - I registri di una interfaccia sono collocati in un *buco di memoria*
  - Un circuito selettore (esterno alla CPU) distingue tra indirizzi di memoria ed indirizzi riservati a interfacce di I/O
  - Le operazioni di lettura e scrittura sui registri di interfaccia si realizzano mediante le istruzioni I/m normalmente utilizzate per i trasferimenti in memoria (MOVE)



- Una operazione di output consiste nel trasferimento di un dato elementare (byte, word, o longword) dalla CPU verso la periferica
- Si sviluppa, per tramite della interfaccia, nei passi seguenti:
  - ① acquisizione (dalla CPU tramite  $I_{CPU-dev}$ ) del dato nel **registro dato**
  - ② acquisizione (dalla CPU tramite  $I_{CPU-dev}$ ) del comando di output nel **registro controllo**
  - ③ trasmissione (tramite  $I_{dev-per}$ ) dei segnali da inviare alla periferica relativi al comando e al dato
  - ④ ricezione (tramite  $I_{dev-per}$ ) di un segnale inviato dalla periferica per segnalare la terminazione della operazione
  - ⑤ modifica del **registro stato** della interfaccia per comunicare alla CPU (tramite  $I_{CPU-dev}$ ) la terminazione della operazione e la disponibilità ad eseguirne un'altra
- I passi 1), 2) e 5) richiedono una interazione con la CPU
- La gestione della interazione CPU-interfaccia è affidata ad un programma: il *driver della periferica*

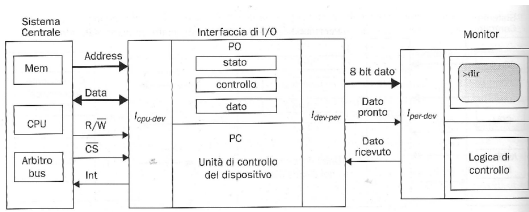


- Una operazione di input consiste nel trasferimento di un dato elementare (byte, word, o longword) dalla periferica verso la CPU
- Si sviluppa, per tramite della interfaccia, nei passi seguenti:
  - ① acquisizione (dalla CPU tramite  $I_{CPU-dev}$ ) del comando di input nel **registro controllo**
  - ② trasmissione (tramite  $I_{dev-per}$ ) dei segnali da inviare alla periferica relativi al comando di input
  - ③ ricezione (tramite  $I_{dev-per}$ ) di un segnale inviato dalla periferica per segnalare la terminazione della operazione
  - ④ acquisizione (tramite  $I_{dev-per}$ ) del dato inviato dalla periferica nel **registro dato**
  - ⑤ modifica del **registro stato** della interfaccia per comunicare alla CPU (tramite  $I_{CPU-dev}$ ) la terminazione della operazione e la disponibilità ad eseguirne un'altra
- I passi 1) e 5) richiedono una interazione con la CPU
- La gestione della interazione CPU-interfaccia è affidata ad un programma: il *driver della periferica*

# Esempio: interfacciamento tra CPU e monitor



- Per illustrare il ruolo del driver, si presenta il collegamento con un monitor a caratteri (periferica di output)
- L'interfaccia è costituita da
  - un registro dato R da 8 bit
  - un registro stato S di 1 bit, messo ad 1 quando la periferica è pronta
  - un registro controllo C di 1 bit, messo ad 1 per indicare la disponibilità di un dato





# Esempio: driver di output

- Driver per un monitor a caratteri (periferica di output)
- Nel driver è realizzato un ciclo di attesa del completamento della operazione (busy-wait) che lo rende inefficiente
- Colloquio CPU-periferica di tipo asincrono: protocollo di handshake

```
driver_monitor(char dato)
{
    // attesa disponibilit  periferica
    repeat
        leggi(S);
    until S=1;
    // inizializzazione bit di controllo
    C = 0;
    // trasferimento del dato nella interfaccia
    R = dato;
    // invio comando di output
    C = 1;
    // attesa segnalazione di completamento operazione (busy-wait)
    repeat
        leggi(S);
    until S=0;
    // ripristino condizioni iniziali
    C = 0;
}
```

# Esempio: driver di output per MC68000



- I registri dell'interfaccia sono collocati nello spazio di indirizzamento della memoria (*I/O memory mapped*)
- Il driver conosce gli indirizzi dei registri della interfaccia
  - IF\_DATA e IF\_CTRL sono registri da 8 bit
- I bit S e C sono il msb (bit 7) ed il lsb (bit 0) di IF\_CTRL

```
        ORG $8020
IF_CTRL DS.B 1
        ORG $8022
IF_DATA DS.B 1
        ORG $8000
% attesa disponibilita periferica
LOOP1   BTST.B #7,IF_CTRL
        BNE LOOP1
% inizializzazione bit di controllo C = 0
        ANDI.B #$FE,IF_CTRL
% trasferimento del dato nella interfaccia
        MOVE.B D0,IF_DATA
% invio comando di output C = 1
        ORI.B #$01,IF_CTRL
% attesa segnalazione di completamento operazione (busy-wait)
LOOP2   BTST.B #7,IF_CTRL
        BEQ LOOP2
% ripristino condizioni iniziali C = 0
        ANDI.B #$FE,IF_CTRL
```



- Al fine di evitare il busy-wait nel driver, si può delegare ad una ISR la gestione della terminazione di una operazione
- Si collega il bit di stato S dell'interfaccia ad una linea di interrupt
- Esempio del monitor a caratteri: driver ed ISR

```
driver_monitor(char dato)
{
    // attesa disponibilita periferica
    repeat
        leggi(S);
    until S=1;
    // inizializzazione bit di controllo
    // abilita interrupt da periferica
    C = 0;
    // trasferimento del dato nella interfaccia
    R = dato;
    // invio comando di output
    C = 1;
    // viene eliminato il busy-wait
}
```

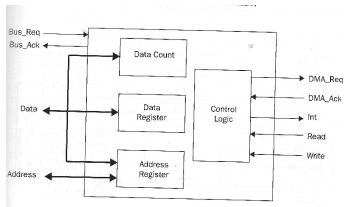
```
ISR_monitor
{
    // Salva il contesto
    ...
    // ripristino condizioni iniziali
    C = 0;
    // disabilita interrupt
    // da periferica
}
```



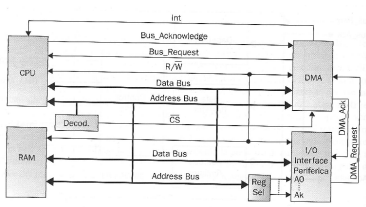
- DMA (*Direct Memory Access*) è un dispositivo capace di operare trasferimenti dati da/verso la memoria senza l'ausilio della CPU
- La CPU può eseguire altre operazioni durante l'operazione di I/O
- CPU e periferica DMA si contendono l'accesso (in lettura/scrittura) alla memoria centrale
- Trasferimenti DMA periferica/memoria o memoria/memoria
- Modi di gestione della concorrenza tra CPU e DMA:
  - Un sistema di arbitraggio decide chi (tra CPU e periferica) ha diritto a fungere da *master* dell'unico bus che collega la memoria
    - Protocollo di handshake
    - linee Bus\_Req (DMA→CPU) e Bus\_Grant (CPU→DMA)
  - La memoria è dotata di bus duali per l'accesso contemporaneo da parte di CPU e DMA
  - La memoria è organizzata gerarchicamente: la CPU opera sulla cache mentre il DMA opera sulla memoria ma occorrono meccanismi che garantiscano la coerenza



- Una interfaccia DMA (*Direct Memory Access*) è un esempio di processore dedicato all'I/O
- Un DMA si impiega di solito per trasferimenti di blocchi di dati che interessano locazioni consecutive di memoria
- Per avviare una operazione DMA occorre una fase preliminare di programmazione del DMA
- Le informazioni minime che la CPU deve scambiare col DMA sono:
  - Indirizzo iniziale e la dimensione del blocco di memoria da caricare nei registri dato
  - La modalità di funzionamento, selezionata tramite un opportuno valore nel registro di controllo
  - L'informazione di terminazione della operazione, acquisita tramite il registro di stato
- La gestione delle operazioni DMA è sincronizzata mediante interruzioni, per cui la terminazione di una operazione determina anche una richiesta di interrupt



- Il DMA pilota una periferica tramite i registri dell'interfaccia di I/O
- DMA e interfaccia collegati allo stesso bus di sistema





- Una volta acquisito il bus, il DMA può trasferire un blocco di dati nei modi seguenti:
  - Una parola alla volta, sottraendo il controllo del bus alla CPU per un solo ciclo (*cycle stealing*)
  - in modo trasparente alla CPU (*hidden*) utilizzando il bus solo nelle fasi del ciclo di istruzioni in cui la CPU non interagisce con la memoria
  - mantenendo il possesso del bus per più cicli consecutivi (*burst mode*) bloccando la CPU