Corso di
Architettura degli Elaboratori
a.a. 2021/2022

Input-Output

Tipi di istruzioni a livello ISA

Istruzioni di Input/Output:

- I/O programmato con busy waiting
- I/O controllato dall'interrupt
- I/O in DMA (Direct Memory Access)

info gen

Comunicazione tra CPU e moduli di I/O

La CPU comunica con i device di I/O tramite i ***controllori***, che hanno il compito di trasformare:

- i comandi della CPU in segnali elettrici per le periferiche
- i segnali delle periferiche in dati per la CPU

Ogni controllore ha al suo interno dei ***registri*** identificati da un indirizzo, che può:

- appartenere allo stesso insieme di indirizzi di memoria (memory mapped I/O)
- essere un indirizzo dedicato (isolated I/O)

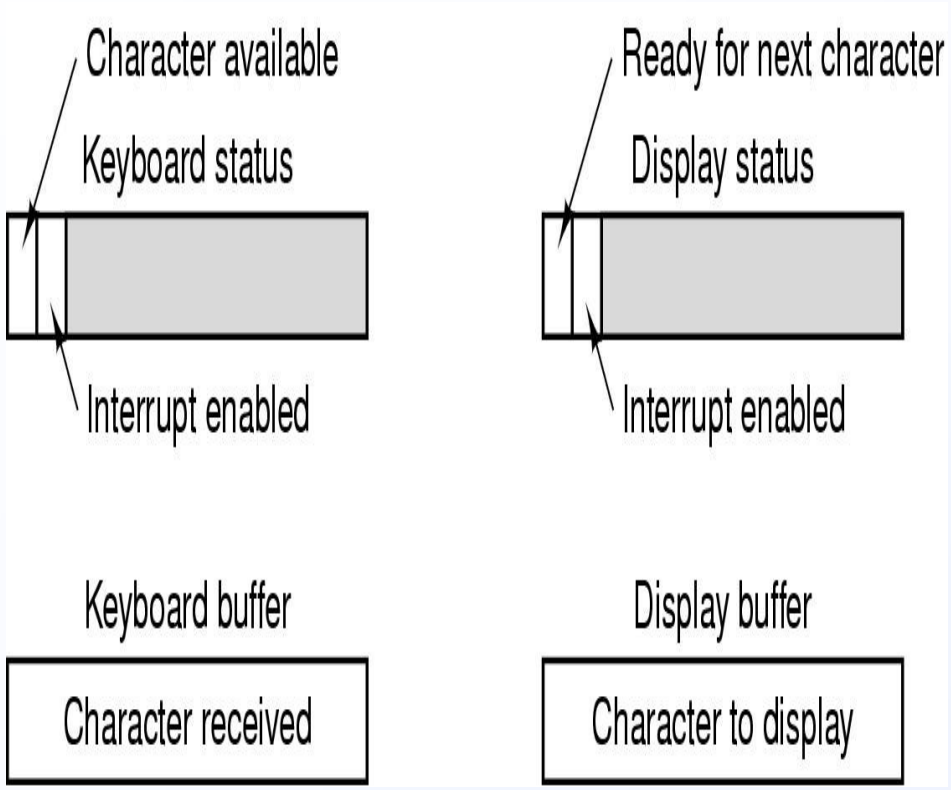
La comunicazione con i controllori è simile agli accessi in memoria

Registri nei controllori

Registri dei controllori contengono:

- *dati* (di ingresso e uscita). Es. per una stampante i dati da stampare)
- *comandi* (dalla CPU alla periferica), ad esempio, i comandi di stampa
- *informazioni sullo stato della periferica* (dalla periferica alla CPU) Es. Informazione sullo stato della stampante o sull'avanzamento dei lavori.

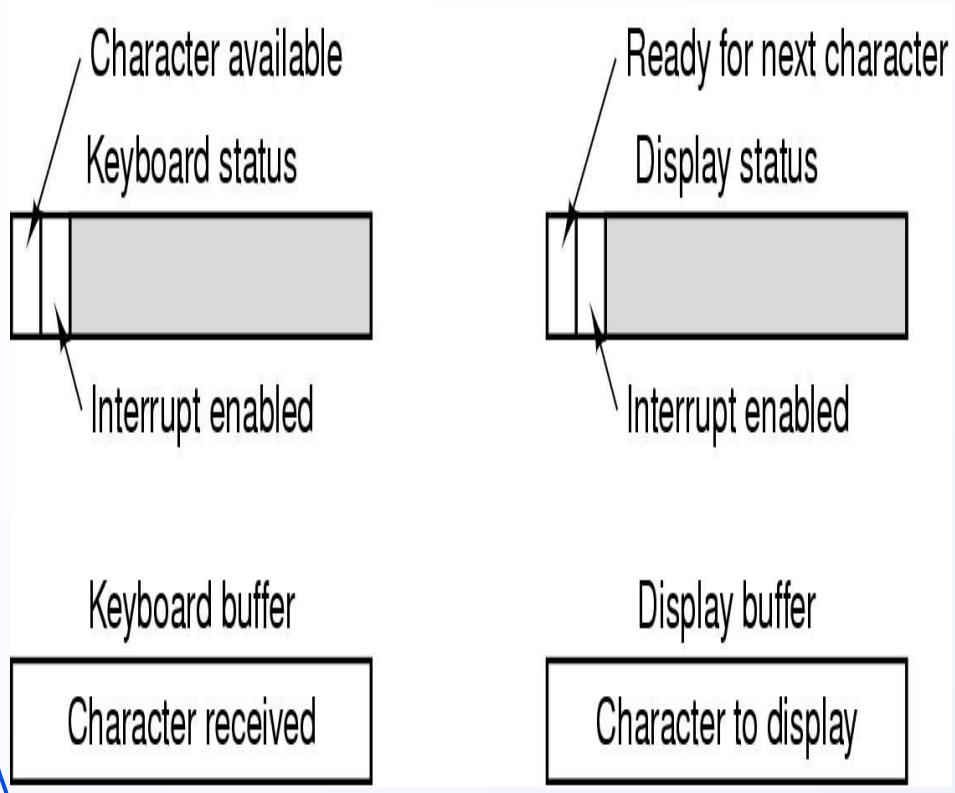
I/O programmato con busy waiting



La CPU controlla lo stato di avanzamento del I/O e lo stato della periferica *ispezionando in un ciclo il bit del registro* di stato del controllore READY fino a che questi non segnala di essere pronto per un nuovo comando di I/O

Svantaggio: la CPU rimane in ciclo attendendo che il dispositivo sia pronto, ***busy waiting***

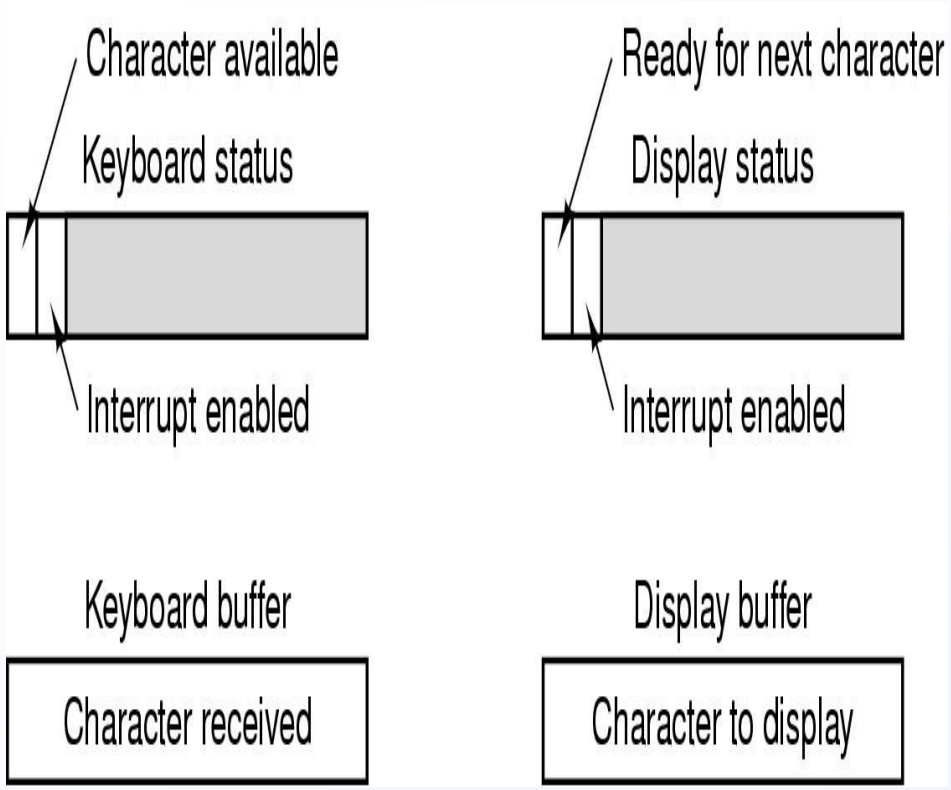
I/O programmato con busy waiting



Lettura es. keyboard:

- Il ***controllore*** della tastiera alla pressione di un tasto mette a 1 un particolare bit del registro di stato e inserisce il codice ASCII del carattere associato al tasto nel registro buffer
- La ***CPU*** legge ripetutatmente il bit del registro di stato fino a quando lo trova uguale a 1
- La ***CPU*** legge il registro buffer, trasferisce il valore per l'elaborazione e azzera il bit del registro di stato

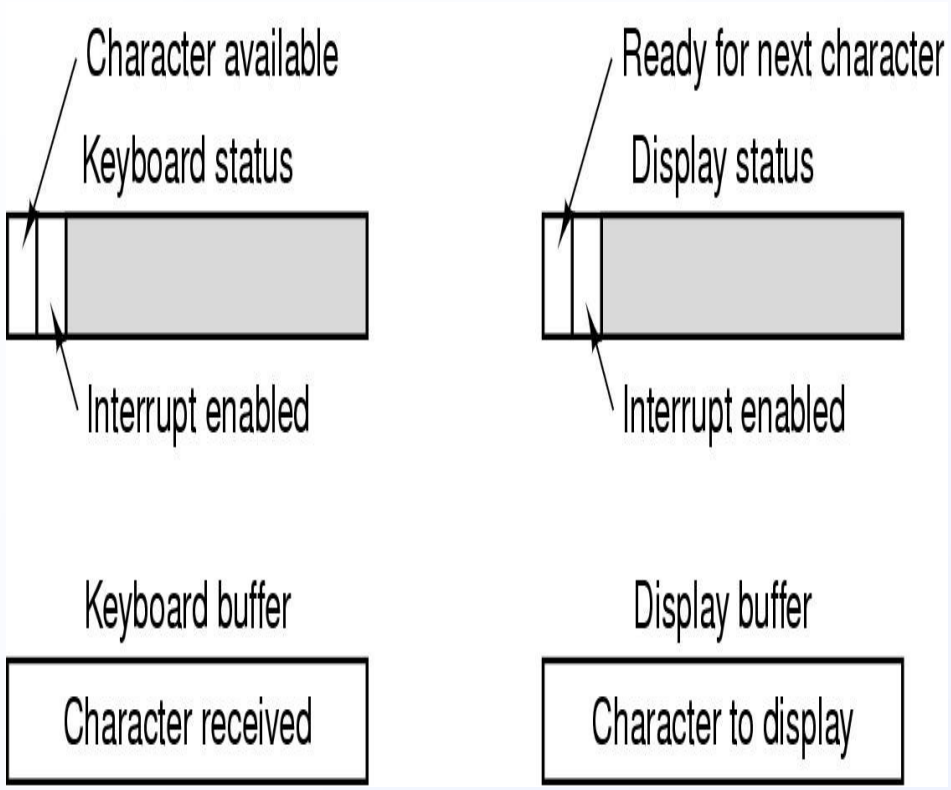
I/O programmato con busy waiting



Scrittura es. display:

- la ***CPU*** attende che il dispositivo sia pronto ad accogliere un carattere (bit Ready = 1)
- la ***CPU*** invia il carattere sul display buffer
- il ***controllore*** azzera il bit Ready e visualizza il contenuto del buffer
- il ***controllore*** segnala di essere pronta a ricevere un nuovo comando (bit Ready = 1)

Interrupt



- Permette di liberarsi dal busy waiting: il *dispositivo* quando ha finito *genera un segnale* che avverte la CPU di aver completato il proprio lavoro (***interrupt***)
- La CPU può abilitare l'interrupt mettendo a 1 un opportuno bit
- **Problema**: serve un'interrupt per ogni carattere letto o scritto!

Interrupt hardware

- Gli interrupt hardware sono cambiamenti nel flusso di controllo legati alla gestione dei periferici e non al programma stesso
- Gli interrupt vanno sempre dai dispositivi alla CPU
- Il dispositivo che segnala un'interruzione asserisce una linea del bus, che è direttamente connessa alla CPU (per esempio a uno o più bit di un registro)
- L'interrupt interrompe il programma in atto e trasferisce il controllo ad un gestore di interrupt che eseguirà le azioni appropriate
- Una volta terminata la gestione dell'interrupt il controllo torna al programma interrotto

Interrupt

- Esempio: scrittura sullo schermo di una riga di “count” caratteri puntata da “ptr”, gestita con interruzione a livello del singolo carattere (il controller del dispositivo è in grado di visualizzare un solo carattere alla volta)
- **AZIONI HW**:
  - Il controller attiva una linea di interrupt del bus di sistema
  - Quando la CPU è pronta a gestire l’interruzione manda un ack sul bus
  - Il controller invia un intero sulle linee dati (vettore di interrupt)
  - La CPU preleva il vettore di interrupt dal bus e lo salva
  - La CPU impila PC ed altri registry di stato sullo stack
  - Il vettore di interrupt è usato come indice di memoria per trovare il valore di PC a cui si trova il gestore di quel tipo di interruzione
  - Si modifica PC, e a volte si evita che un’altra interruzione possa interrompere la gestione dell’interruzione in corso di esecuzione

Interrupt

AZIONI SW:

- Il codice ISA di gestione delle interruzioni salva tutti i registri (stato del programma che è stato interrotto)
- Si possono leggere tutte le informazioni dal buffer del dispositivo
- Se c'è stato un errore di I/O lo si gestisce
- Le variabili ptr e count vengono aggiornate, se ci sono altri caratteri da visualizzare si copia il carattere puntato da ptr nel buffer del controller del dispositivo
- Eventuali altri ack di fine trattamento interruzione
- Ripristino registri
- Esecuzione di una apposita istruzione di "return from interrupt – RETI" che ripristina modalità e stato della CPU.

Interrupt

- Gli interrupt sono ***asincroni*** rispetto al programma

- Gli interrupt devono essere gestiti in modo ***trasparente***: lo stato dell'esecuzione dopo la gestione dell'interrupt deve tornare esattamente come era prima dell'interrupt stesso

Interrupt

E se ci sono *più dispositivi* che lavorano su interruzione?

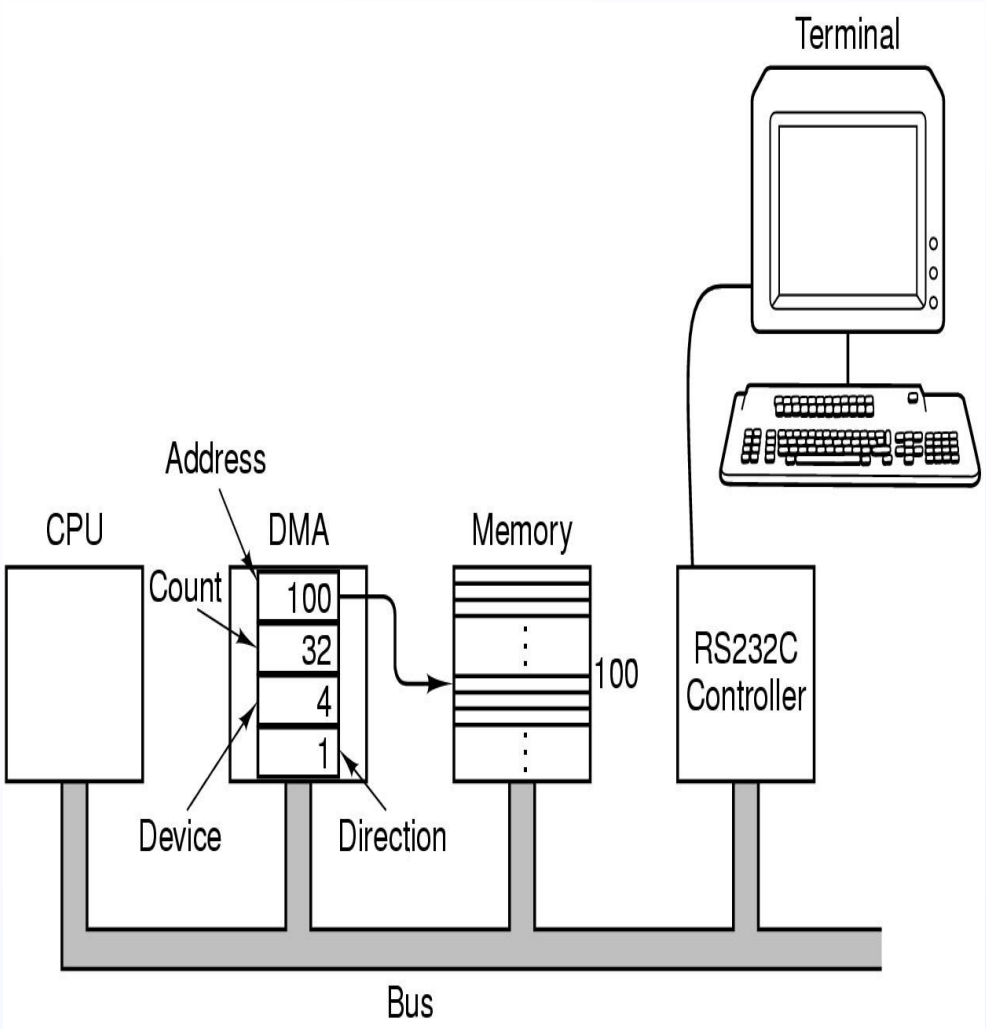
- Il gestore delle interruzioni disabilita le interruzioni
- Oppure, si definiscono delle priorità tra i dispositivi che possono generare delle interrupt e si permette che una sopravvenuta interruzione interrompa la gestione dell'interrupt precedente

- La priorità è proporzionale alla criticità del dispositivo che solleva l'interrupt
- Ad ogni dispositivo è assegnato un livello di priorità e un tipo di interruzione (**mascherabile** o **non mascherabile**)
- Se la CPU non permette più livelli di priorità si utilizza normalmente un chip apposito che gestisce in modo "autonomo" le interruzioni

Altri tipi di cicli di bus

- Ciclo di bus per il segnale di **interrupt**. Segnale di interrupt viene inviato dai periferici di I/O quando hanno terminato un lavoro. Se più periferici devono segnalare contemporaneamente esiste un problema analogo all'arbitraggio del bus. Priorità negli interrupt.

DMA (Direct Memory Access)



- I/O programmato ma svolto da un'apposita componente con accesso diretto al bus
- Il DMA è impostato direttamente dal software o il sistema operativo inizializzando opportuni registri
- La CPU e il DMA si contendono l'uso del bus

Trap (o eccezione o interrupt sincroni)

- Sono chiamate a procedure automatiche che possono essere causata dal verificarsi di qualche condizione eccezionale oppure da istruzioni apposite per richiedere servizi di base del sistema operativo
- Gestore delle trap: procedura di gestione della richiesta di trap la cui posizione (indirizzo) è memorizzata in una locazione fissa a cui il programma salta in caso di trap
- L'eventuale eccezione è individuata dall'hardware
- Es. gestione dell'overflow (il test è fatto via hardware) in parallelo con l'esecuzione di altre funzionalità
- Le trap sono ***sincrone*** al programma

Trap (o eccezione)

Esistono due modi diversi di gestione dell'overflow:

- *Senza trap*: l'hw setta un bit in un apposito registro e il programmatore ISA, se lo desidera, dopo un'istruzione che può causare overflow, testa il registro e salta ad apposita procedura
- *Con trap*: la condizione di overflow (rilevata da hw) genera una trap, che modifica il flusso di controllo ad una locazione prefissata (gestore della trap)

***Differenze**:* nel primo caso ci sono molte più istruzioni ISA per il controllo dell'overflow sparse nel programma, nel secondo caso c'è una modifica del microinterprete che alla fine dell'esecuzione delle istruzioni aritmetiche, controlla l'overflow