Sistemi Operativi

Ionut Zbirciog

14 December 2023

Principi dell'Hardware di I/O

Dispositivi di I/O

I dispositivi di I/O si possono suddividere in:

- **Dispositivi a blocchi:** Archiviano informazioni in blocchi di dimensioni fisse, ognuno con il proprio indirizzo. Tutti i trasferimenti sono in unità di uno o più blocchi interi. Caratteristica principale: ogni blocco è indipendente dagli altri. Esempi di dispositivi a blocchi sono dischi fissi magnetici, SSD, ecc.
- **Dispositivi a caratteri:** Rilasciano o accettano un flusso di caratteri senza alcuna struttura a blocchi. Non sono indirizzabili e non hanno operazioni di ricerca. Esempi di dispositivi a caratteri sono stampanti, mouse, tastiere, ecc.

Questa classificazione, tuttavia, non include tutti i dispositivi di I/O, come ad esempio clock, schermi o touchscreen.

Dispositivo	Velocità di trasferimento
Tastiera	10 byte/s
Mouse	100 byte/s
Modem a 56 K	7 KB/s
Bluetooth 5 BLE	256 KB/s
Scanner a 300 dpi	1 MB/s
Videocamera digitale	3,5 MB/s
Wireless 802.11n	37,5 MB/s
USB 2.0	60 MB/s
Disco Blu-ray 16x	72 MB/s
Gigabit Ethernet	125 MB/s
Disco fisso SATA 3	600 MB/s
USB 3.0	625 MB/s
Bus PCIe 3.0 single lane	985 MB/s
Wireless 802.11ax	1,25 GB/s
SSD NVME PCIe Gen 3.0 (lettura)	3,5 GB/s
USB 4.0	5 GB/s
PCI Express 6.0	126 GB/s

Figure 1: Dispositivi di I/O

Controller dei Dispositivi

I dispositivi di I/O sono composti da una parte elettronica chiamata *controller del dispositivo* e da una parte meccanica, ovvero il dispositivo stesso. Il controller è presente nei computer come un chip o una scheda a circuiti stampati. Molti controller possono gestire diversi dispositivi identici. Le interfacce fra il controller e il dispositivo possono essere standardizzate come ANSI, IEEE, ISO o de facto, come SATA, SCSI, USB o Thunderbolt.

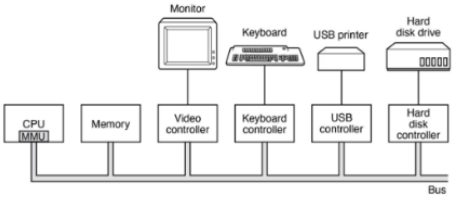

Figure 2: Dispositivi e vari controller

L'interfaccia tra il controller e il dispositivo è di livello molto basso; il compito del controller è convertire il flusso seriale di bit in blocchi di byte, correggere errori e trasferire in memoria centrale. Senza il controller, il programmatore dovrebbe gestire dettagli complessi, come la modulazione di ciascun pixel. Il controller viene inizializzato dal sistema operativo con parametri essenziali, poi viene delegato alla gestione dei dettagli complessi.

Dalla Porta Parallela alla Porta USB

Porta Parallela: Tipo di interfaccia di comunicazione tra computer e dispositivi. Trasmette dati multi-bit simultaneamente su più canali. Comunemente dotata di 25 o 36 pin.

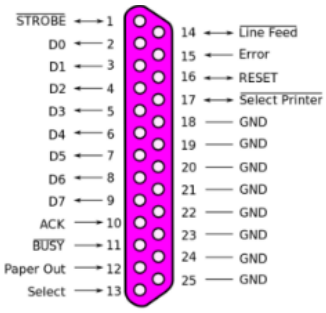

Figure 3: Porta Parallela

- Pin 1-8: Trasferimento di dati
- Pin 9-16: Controlli e status
- Pin 17-25: Masse e alimentazione

Porta USB: Universal Serial Bus è un'interfaccia standardizzata per la connessione e comunicazione tra dispositivi e computer. È anche utilizzata per fornire alimentazione elettrica oltre che dati. È ampiamente utilizzata in vari dispositivi come smartphone, periferiche computer e dispositivi di archiviazione. È facile da usare, connessione plug-and-play.

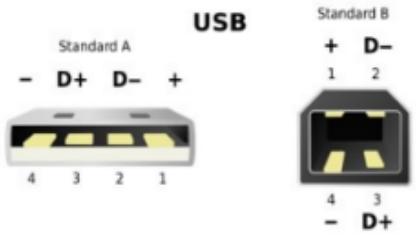

Figure 4: Porte USB

- Pin 1: Vcc, alimentazione 5V
- Pin 2: D-, Dati negativi
- Pin 3: D+, Dati positivi
- Pin 4: GND, Terra

I/O Mappato in Memoria

Ogni controller dispone di alcuni registri usati per le comunicazioni con la CPU. Scrivendo in questi registri, il sistema operativo può ordinare al dispositivo di inviare dati, accettarli, accendersi e spegnersi o eseguire qualche altra azione. Leggendo da questi registri, il sistema operativo apprende quale sia lo stato del dispositivo, se sia pronto ad accettare un nuovo comando e così via.

Oltre ai registri di controllo, molti dispositivi hanno un buffer di dati su cui il sistema operativo può scrivere e leggere. Ad esempio, un metodo classico dei computer per visualizzare i punti sullo schermo è avere una RAM video.

Port-Mapped I/O

Questo metodo assegna a ogni registro di controllo un numero di porta di I/O associato, di solito un intero di 8 o 16 bit. Per esempio, per leggere da una porta: `IN REG, PORT`, mentre per scrivere: `OUT PORT, REG`.

È importante avere una separazione degli spazi di indirizzi della memoria e dell'I/O. Quindi `IN R0, 4` legge dalla porta 4 e salva in R0, mentre `MOV R0, 4` legge dall'indirizzo di memoria 4 e salva in R0. I due "4" sono indirizzi diversi.

Memory-Mapped I/O

Questo metodo assegna a ogni registro di controllo un indirizzo di memoria univoco. I registri di controllo sono mappati nello spazio di memoria. Viene eliminata la necessità di istruzioni speciali di I/O come IN e OUT. Inoltre, i registri di controllo possono essere trattati come variabili in un

linguaggio come C, facilitando la scrittura di driver direttamente in C e non in assembly. Non c'è bisogno di una protezione complicata; il sistema operativo semplicemente non deve mettere questi indirizzi nello spazio degli indirizzi virtuali di qualunque processo utente. Attraverso la gestione delle pagine di memoria, è possibile dare un controllo selettivo su dispositivi specifici. Consente l'esecuzione di driver di dispositivi in spazi separati, aumentando la sicurezza e riducendo le dimensioni del kernel. Uno svantaggio di questo metodo è la gestione inefficiente delle cache e dei registri mappati in memoria.

Un possibile approccio ibrido è filtrare gli indirizzi per distinguere tra memoria e dispositivi di I/O. Gli indirizzi di I/O vengono inoltrati ai dispositivi anziché alla memoria.

Direct Memory Access (DMA)

Il DMA permette alla CPU di scambiare dati con i controller dei dispositivi bypassando il trasferimento manuale byte per byte, riducendo lo spreco di tempo alla CPU e migliorando l'efficienza del trasferimento dei dati.

Confrontiamo ora il trasferimento di dati tradizionale, ovvero senza DMA, e il trasferimento di dati con DMA.

- **Senza DMA:** Il controller del disco legge i dati e li memorizza nel suo buffer. Dopo aver controllato gli errori, provoca un interrupt e il sistema operativo copia i dati in memoria, sprecando tempo alla CPU.
- **Con DMA:**
  - Passo 1: La CPU imposta il controller DMA e invia un comando al controller del disco.
  - Passo 2: Il controller DMA richiede la lettura al controller del disco.
  - Passo 3: Scrittura in memoria da parte del controller del disco.
  - Passo 4: Conferma dal controller del disco al DMA.

I passi 2-4 vengono ripetuti fino al completamento del trasferimento, al termine il DMA invia un interrupt alla CPU. In questo modo, la CPU delega il trasferimento dei dati al DMA, permettendo alla CPU di eseguire altre operazioni.

Modalità DMA e interazioni con il BUS:

- **Cycle Stealing:** DMA trasferisce una parola per volta, "rubando" cicli alla CPU nel caso in cui anche alla CPU servisse il BUS.
- **Modalità Burst:** DMA ottiene il controllo completo del bus, eseguendo trasferimenti multipli in una volta.
- **Fly-by Mode:** DMA trasferisce dati direttamente alla memoria principale senza intermediari.

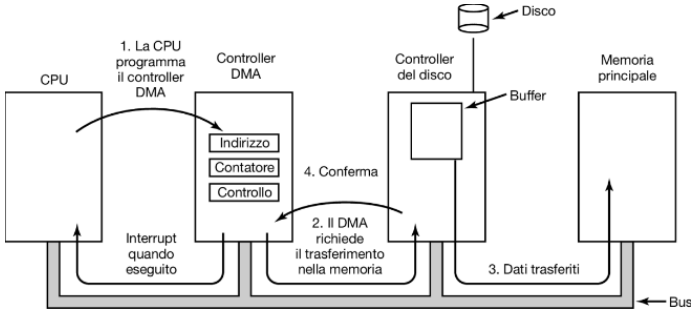

Figure 5: Funzionamento DMA

Interrupt

Gli interrupt possono essere causati da diverse cause:

- **Trap:** Azione deliberata da parte del codice del programma, ad esempio una trap nel kernel per una chiamata di sistema.
- **Fault o Eccezione:** Azioni non deliberate, come errori di segmentazione o divisione per 0.
- **Interrupt Hardware:** Segnali inviati da dispositivi come stampanti o reti della CPU.

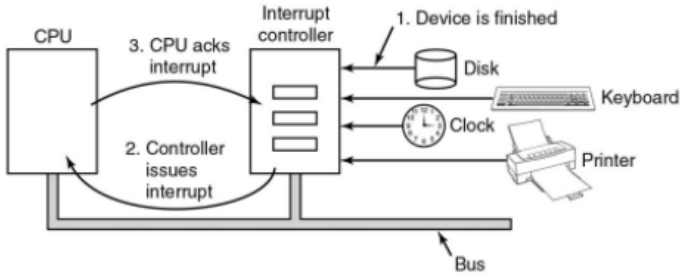

Figure 6: Cause dell'Interrupt

Un dispositivo di I/O invia un segnale di interrupt alla CPU mediante un bus, poi viene gestito dal chip nel controller degli interrupt sulla scheda madre. Se non ci sono altri interrupt in corso, il controller gestisce immediatamente l'interrupt; in caso di interrupt simultanei, il dispositivo è temporaneamente ignorato.

Processo di Gestione degli Interrupt

- **Segnalazione dell'Interrupt alla CPU:** Il controller assegna un numero alle linee degli indirizzi per specificare il dispositivo che richiede attenzione e invia un segnale di interruzione alla CPU.
- **Interruzione e Gestione da Parte della CPU:** La CPU interrompe il suo attuale compito. Utilizza un numero sulle linee degli indirizzi come indice nella tabella del vettore degli interrupt per ottenere un nuovo contatore di programma.
- **Il Vettore degli Interrupt:** Punta all'inizio della procedura di servizio degli interrupt corrispondente.
- **Conferma e Gestione degli Interrupt:** La procedura di servizio conferma l'interrupt scrivendo su una porta del controller degli interrupt.
- **Salvataggio dello Stato:** Al minimo, il contatore di programma deve essere salvato per riavviare i processi interrotti. Il salvataggio avviene nei registri interni o sullo stack. La CPU deve decidere se usare lo stack corrente del processo rischiando errori di pagina e puntatori illeciti o lo stack del kernel, producendo un overhead, poiché deve cambiare il contesto della MMU e possibile invalidazione della cache o del TLB.

Tipologie di Interrupt

1. **Interrupt Precisi:** Situazione in cui il sistema può determinare con esattezza quali istruzioni sono state completate al momento dell'interrupt e quali no. In questo caso, il PC è salvato in un luogo noto. Tutte le istruzioni eseguite prima del PC sono completate. Nessuna istruzione dopo il PC è stata eseguita. Lo stato dell'istruzione puntata dal PC è noto. Per gestire questo interrupt, la CPU cancella gli effetti di eventuali istruzioni transitorie eseguite dopo il PC, garantendo compatibilità e prevedibilità.
2. **Interrupt Imprecisi:** Condizione in cui diverse istruzioni vicino al contatore di programma si trovano in vari stati di completamento al momento dell'interrupt, rendendo incerto lo stato esatto del programma. Richiede che la CPU salvi una grande quantità di stato interno sullo stack, rendendo tutto il processo di gestione molto lento.