
Sistemi di INPUT/OUTPUT

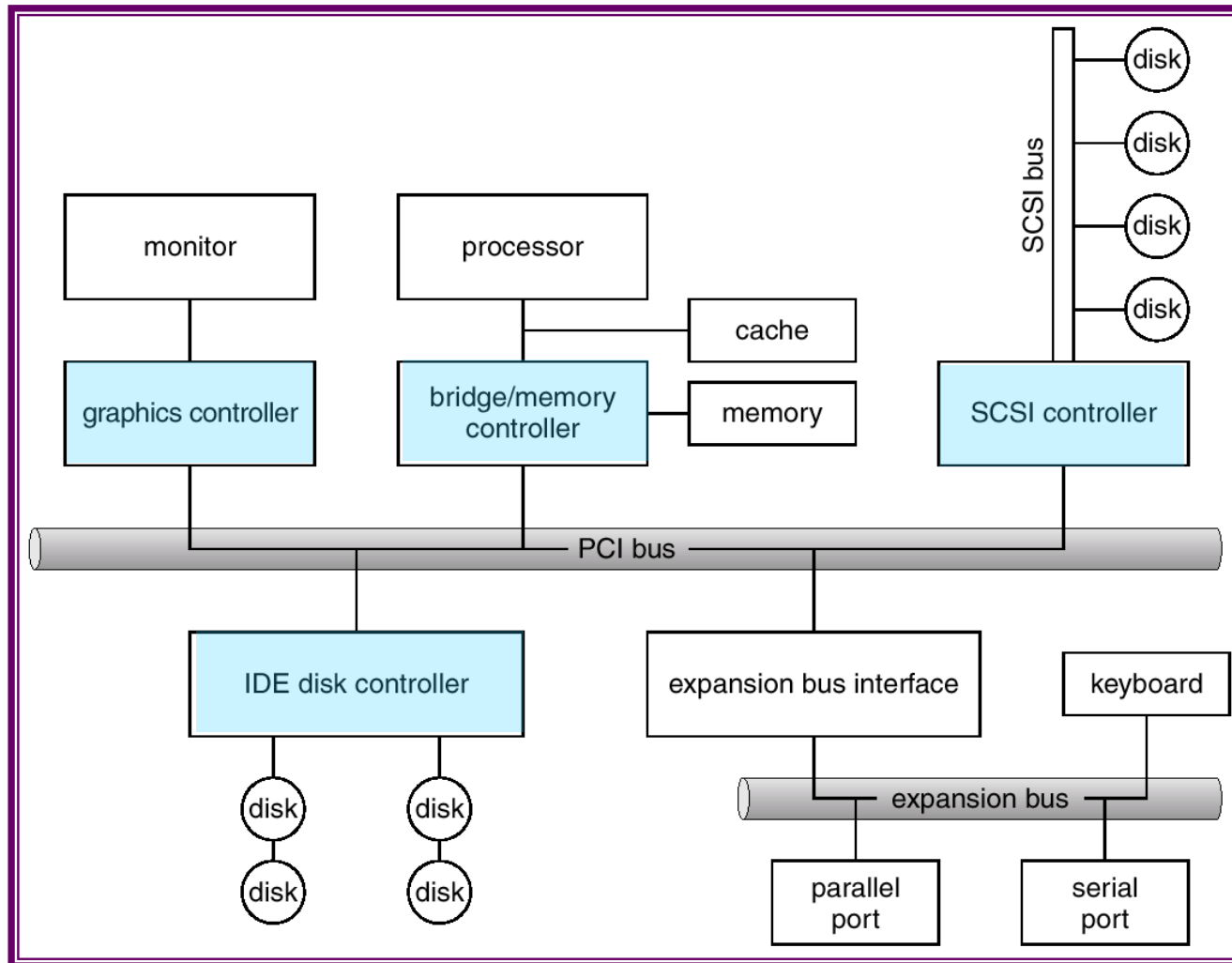
Sistemi I/O

- Hardware di I/O
- Interfaccia di I/O per le applicazioni
- Sottosistema per l'I/O del kernel
- Trasformazione delle richieste di I/O
- Prestazioni

I/O Hardware

- Grande varietà di dispositivi di I/O ➔ molti metodi di gestione.
- Sottosistema di I/O = {metodi di gestione dell'I/O}
- Concetti comuni:
 - Porte
 - Bus (*daisy chain* o accesso diretto condiviso)
 - Controller (adattatore host)
- I dispositivi hanno degli indirizzi che possono essere usati tramite due metodi:
 - Istruzioni di I/O e registri di I/O (status, control, data-in, data-out)
 - I/O mappato in memoria.

Struttura del Bus di un PC



Indirizzi dei dispositivi di I/O sui PC

I/O address range (hexadecimal)	device
000-00F	DMA controller
020-021	interrupt controller
040-043	timer
200-20F	game controller
2F8-2FF	serial port (secondary)
320-32F	hard-disk controller
378-37F	parallel port
3D0-3DF	graphics controller
3F0-3F7	diskette-drive controller
3F8-3FF	serial port (primary)

Se si usa l'I/O mappato in memoria i driver dei diversi dispositivi usano delle aree di memoria in cui memorizzare i dati che usano.

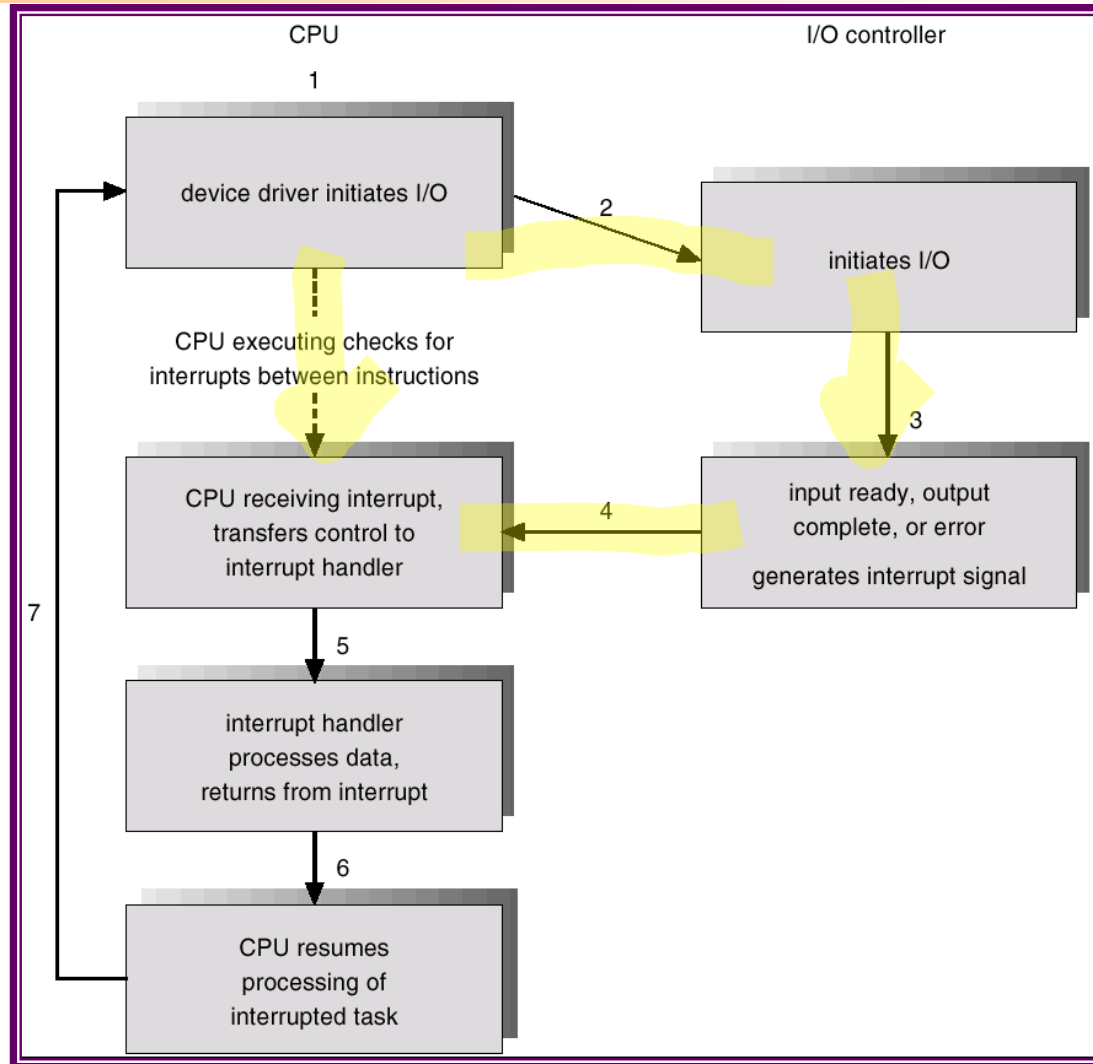
Interrogazione ciclica (Polling)

- L'interazione tra un controller e la CPU avviene secondo il modello produttore-consumatore.
- Per determinare lo stato del dispositivo si usano tre bit principali del registro di stato del dispositivo:
 - command-ready
 - busy
 - error
- La CPU esegue una interrogazione ciclica (*Busy-wait cycle*) ponendosi in attesa che il bit busy sia 0.
- Questo provoca una attesa attiva della CPU.

Interruzione (Interrupt)

- CPU ha una linea di richiesta di interrupt per ricevere un segnale dal dispositivo quando è libero. (attesa passiva)
- La CPU invoca una routine di gestione all'arrivo dell'interruzione.
- Esiste anche una linea degli interrupt “mascherabili” per evitare di ricevere interruzioni in particolari condizioni.
- Il vettore degli interrupt serve per invocare il gestore opportuno per ogni interrupt ricevuto.
 - Basato su priorità
 - alcuni interrupt non sono “mascherabili”.
- Il meccanismo degli interrupt è anche usato per le eccezioni.

Ciclo di I/O basato su interrupt



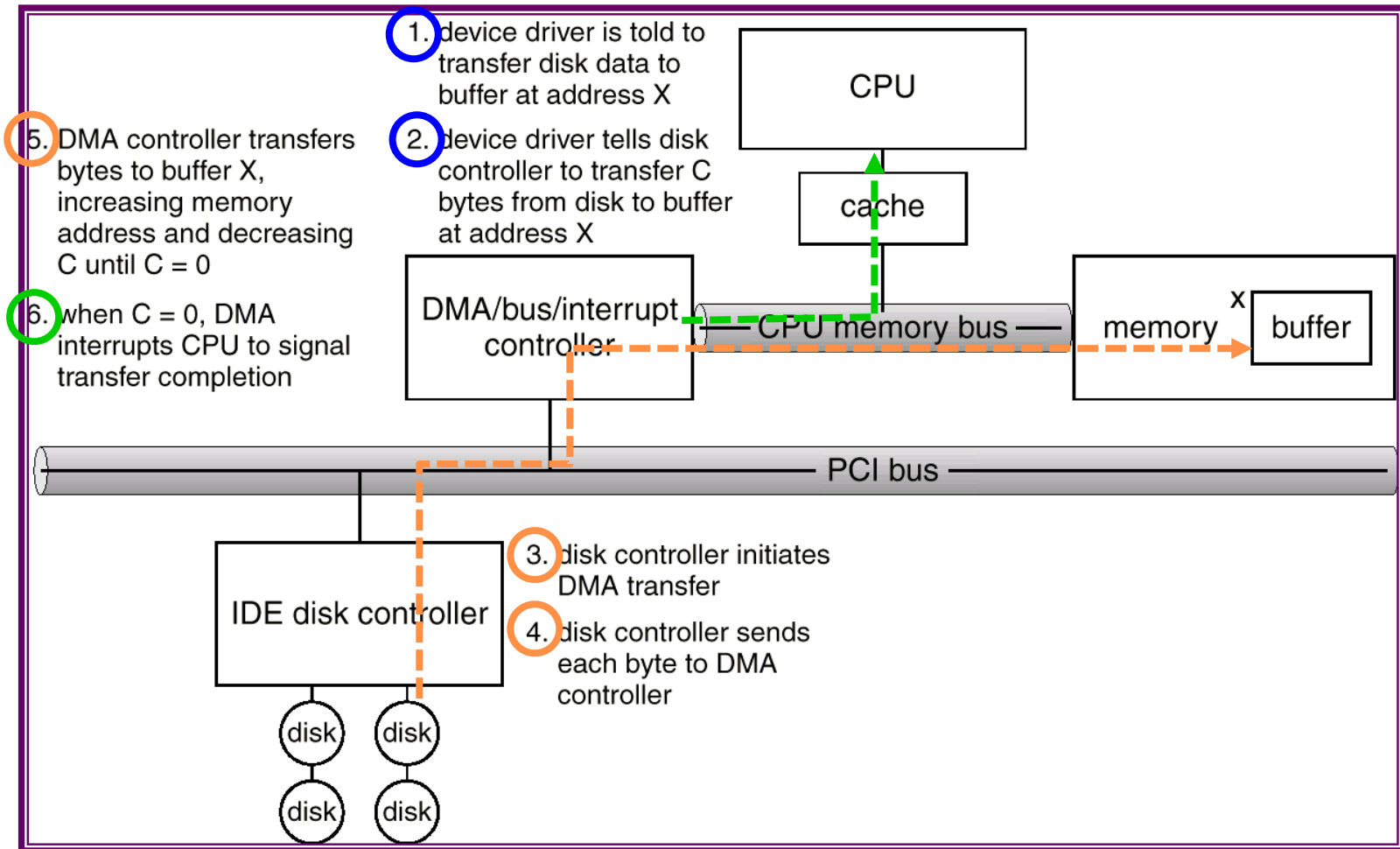
Vettore degli interrupt dell' Intel Pentium

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access (DMA)

- DMA: processore che ha lo scopo di effettuare il controllo nel trasferimento di dati.
- Usato per liberare la CPU dai compiti di controllo.
- Richiede un controller DMA.
- L'interazione (*handshaking*) tra il controller DMA e il controller di un dispositivo realizza il trasferimento.

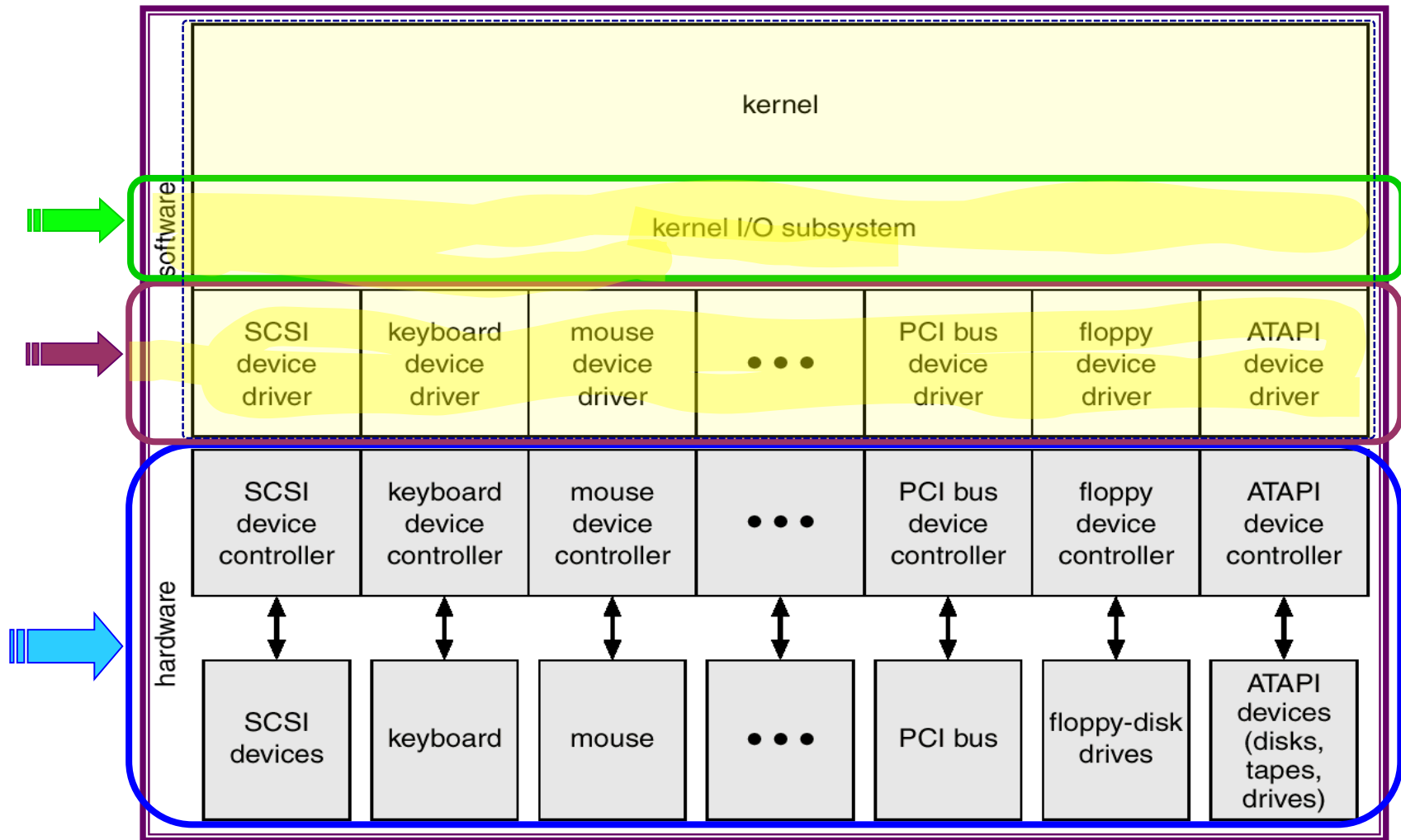
Passi di un trasferimento con DMA



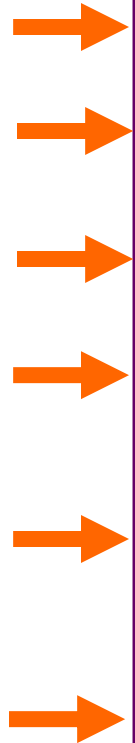
Interfaccia di I/O per le Applicazioni

- Le system call di I/O offrono una interfaccia uniforme ai diversi comportamenti dei dispositivi.
- I driver dei dispositivi (*device-driver*) sono la parte del sistema operativo che permette di usare l'I/O in maniera uniforme (nascondendo le differenze).
- I dispositivi variano in molti aspetti
 - a caratteri o a blocchi,
 - ad accesso sequenziale o diretto,
 - condivisi o dedicati
 - con diverse velocità
 - lettura, scrittura o entrambi.

Struttura del kernel per I/O



Caratteristiche dei dispositivi di I/O



aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read&write	CD-ROM graphics controller disk

Dispositivi a caratteri o a blocchi

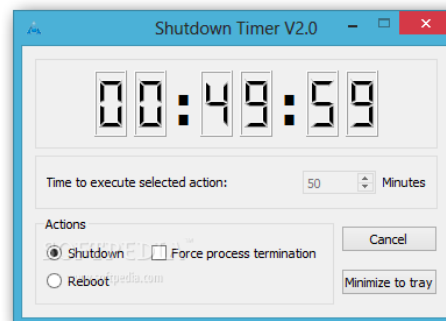
- I dispositivi a blocchi includono i dischi:
 - operazioni: lettura, scrittura, posizionamento
 - I/O di basso livello e accesso al file system
 - accesso ai file mappato in memoria.
- I dispositivi a caratteri includono la tastiera, il mouse, le porte seriali:
 - operazioni: get, put
 - si possono costruire operazioni più strutturate: su linee, su buffer di caratteri.

Dispositivi di rete

- Diversi dai dispositivi a caratteri o a blocchi.
- Unix e Windows NT/9x/2000 includono una *socket interface*
 - Separa i protocolli di rete dalle operazioni di rete
 - Include l'operazione di select per controllare più porte.
- Esistono diversi molti modi di uso dei dispositivi di rete (*pipes*, code FIFO, *stream*, *mailbox*)

Clock e Timer

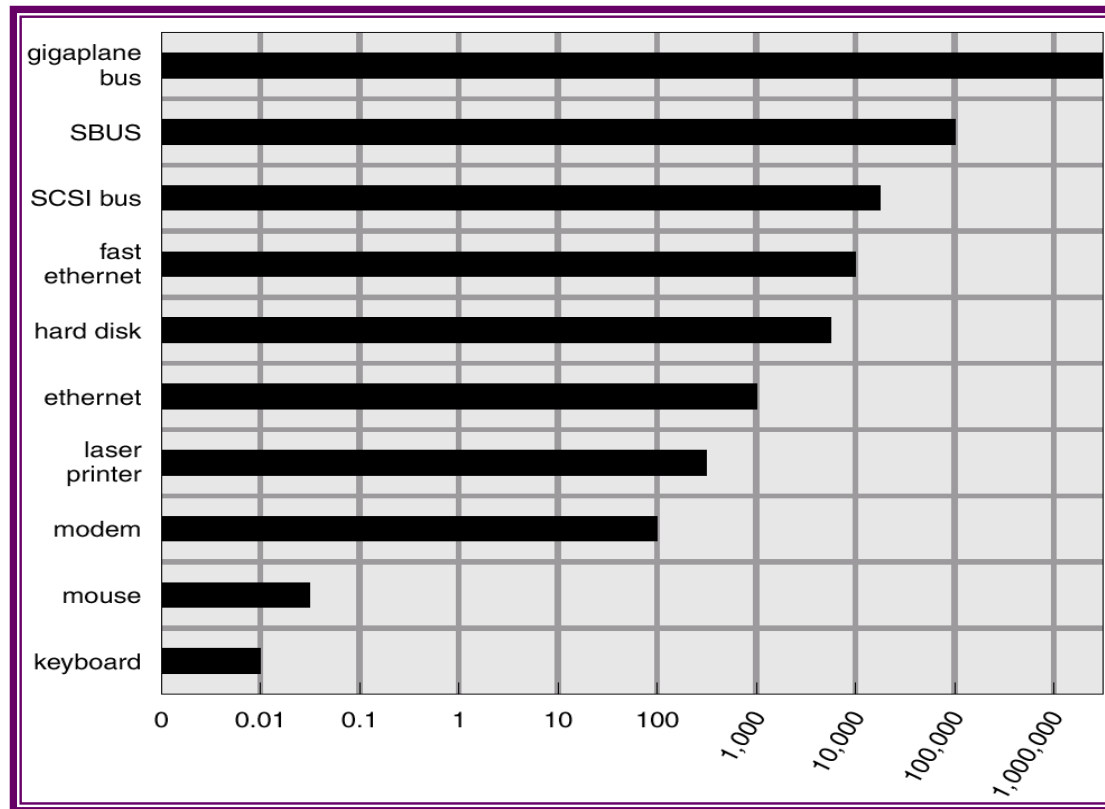
- Forniscono l'ora corrente, il tempo trascorso da un dato evento, la regolazione del timer per indicare quando avviare un'operazione.
- Sistemi *timer programmabili* permettono di avviare operazioni o interrupt in dati intervalli temporali.
- In UNIX la system call ***ioctl*** nelle sue diverse forme permette di gestire operazioni con timer.



Sottosistema per l'I/O del Kernel

- Il kernel offre un insieme di servizi per le operazioni di I/O.
- **Scheduling**
 - Alcune richieste di I/O sono ordinate tramite le code dei dispositivi.
 - Alcuni S.O. usano delle politiche *fair* (eque).
- **Buffering:** memorizza dati in memoria centrale durante il loro trasferimento tra dispositivi:
 - tra dispositivi con velocità diverse (tastiera-disco)
 - tra dispositivi con dimensioni di memoria diversa
 - per realizzare la “semantica di copia”.

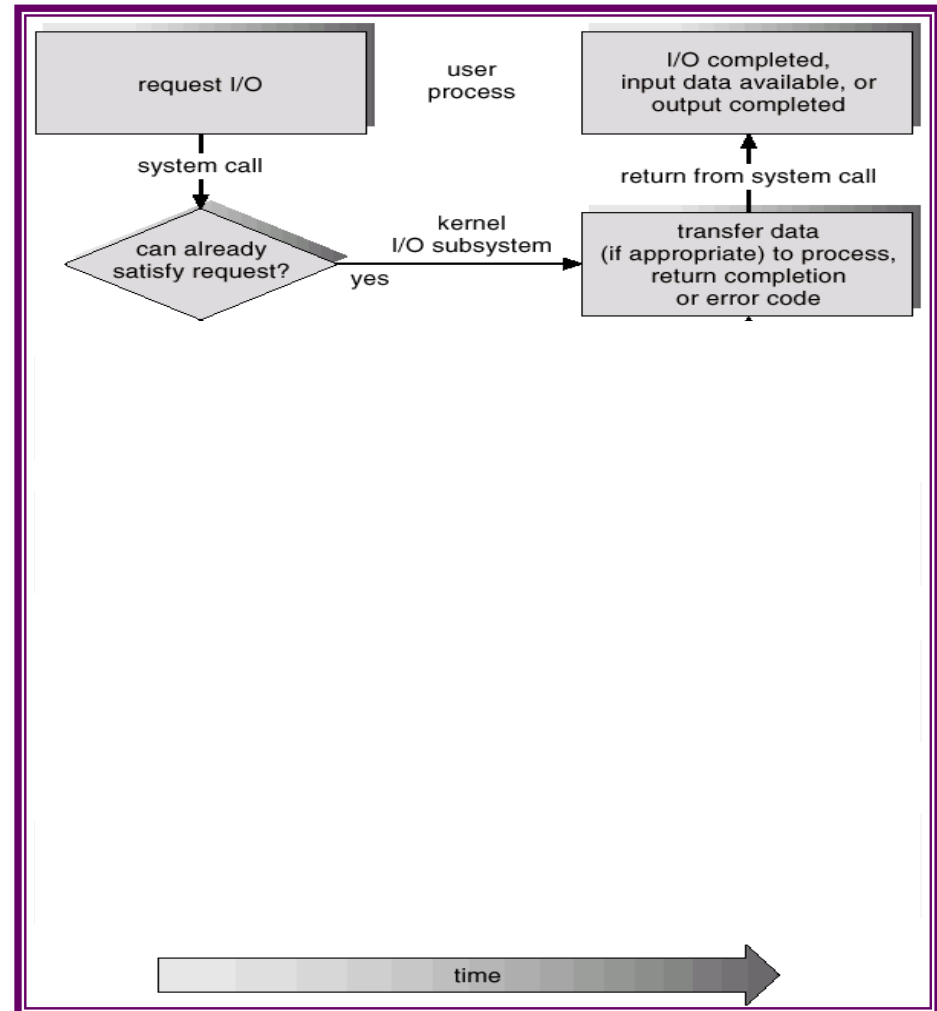
Velocità di trasfer. dei dispositivi sul Sun Enterprise



L'uso del buffer in memoria è utile per la comunicazione tra dispositivi con velocità diverse

Schema di esecuzione di una richiesta di I/O

Sequenza di operazioni svolte dai diversi componenti hw/sw per la gestione di un'operazione di lettura bloccante



Prestazioni

- L'uso dell'I/O è uno dei fattori che contribuiscono maggiormente a determinare le prestazioni delle applicazioni. Questo perché:
 - Richiede tempo di CPU per eseguire i driver
 - Si hanno diversi context switch a causa degli interrupt
 - Serve uno scheduling equo per i processi che effettuano operazioni di I/O
 - Si hanno molti trasferimenti di dati
 - Si genera traffico di dati.

Migliorare le prestazioni

- Per un'esecuzione efficiente delle operazioni di I/O è necessario:
 - ridurre il numero di context switch
 - ridurre la copia di dati
 - ridurre gli interrupt usando trasferimenti di grandi quantità di dati in un'unica operazione, controller intelligenti, polling.
 - usare il DMA
 - bilanciare il carico di CPU, memoria, bus, e sistema di I/O.