Gestione dei dispositivi periferici (I/O) e della memoria secondaria

Gestione delle periferiche di I/O e della memoria secondaria

- I due compiti principali di un computer sono l'elaborazione e l'I/O
  - In molti casi, il compito principale è l'I/O, e l'elaborazione è meramente incidentale
  - Ad esempio, quando si consulta una pagina Web o si modifica un file, ciò che interessa l'utente è leggere o inserire alcune informazioni, non elaborare una qualche risposta
- Il ruolo del SO nell'I/O del calcolatore consiste nel gestire e controllare le operazioni e i dispositivi di I/O
- Il codice per l'I/O
  - rappresenta una parte significativa del SO
  - è fondamentale per l'integrazione della grandissima varietà di dispositivi esistenti nei calcolatori e nei SO moderni

Obiettivi

- Illustrare i principi alla base dell'HW di I/O e le problematiche correlate
- Descrivere l'uso dei registri di un controllore di dispositivo e i metodi di accesso relativi
- Descrivere la struttura del sottosistema di I/O di un SO
- Discutere i servizi di I/O forniti dal SO
- Spiegare la trasformazione delle richieste di I/O in operazioni HW
- Descrivere le principali caratteristiche dei dischi rigidi
- Spiegare la gestione dei dischi rigidi
- Illustrare gli algoritmi di scheduling delle richieste ai dischi rigidi

Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

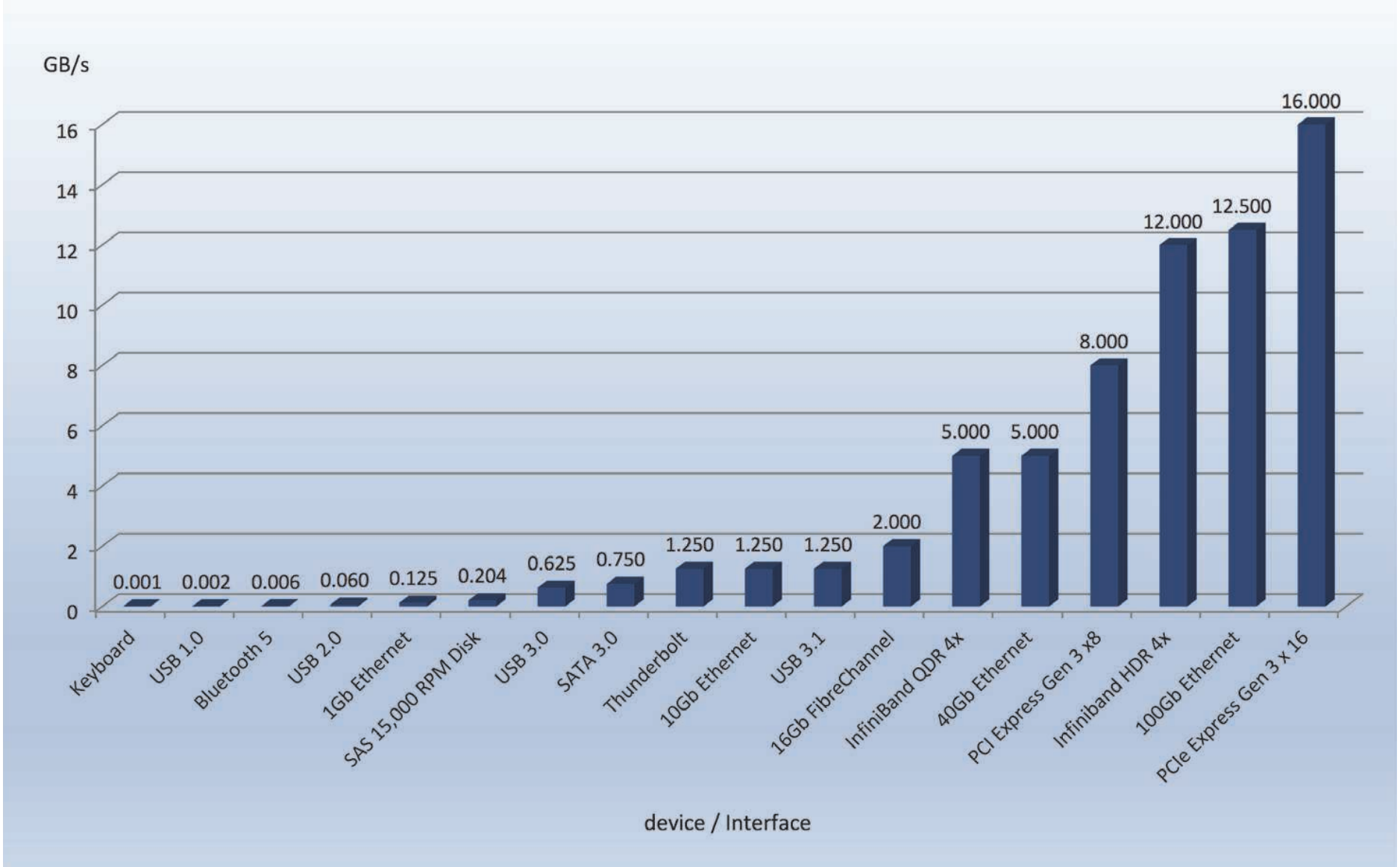
Dispositivi di I/O

- I computer gestiscono una enorme varietà di dispositivi (o periferiche) di I/O, es.
  - dispositivi di archiviazione (dischi, nastri)
  - dispositivi di trasmissione (connessioni di rete, Bluetooth)
  - dispositivi di interfaccia persona-macchina (schermi, tastiere, mouse, ingressi e uscite audio)
  - dispositivi specializzati, es. quelli coinvolti nella guida di un jet
  - …
- Pertanto, i dispositivi di I/O possono differire in molti aspetti

Aspetti dei dispositivi di I/O

- Numero e tipo di funzioni che possono svolgere
- Quantità di errori e malfunzionamenti che possono generare
- Sorgente o destinazione dei dati:
  - operatore umano (terminale video, tastiera, mouse, stampante, ...)
  - altre apparecchiature (sensori, attuatori, dischi, nastri, ...)
  - apparecchiature o utenti remoti (schede di rete, modem, ...)
- Modalità d'accesso: sequenziale (trasferisce dati in un ordine fisso determinato dal dispositivo) o diretto/casuale (l'utente può richiedere l'accesso ad una qualsiasi delle locazioni disponibili)
- Condivisione: come qualsiasi risorsa, possono essere *dedicati* ad un processo per volta o essere *condivisi* tra più processi o thread
- Direzione dell'I/O: input e output, solo input, solo output
- Modalità di I/O: I/O sincrono o I/O asincrono
- Velocità di trasmissione dei dati (da pochi byte al secondo a diversi gigabyte al secondo)
- Organizzazione/trasferimento dei dati: a blocchi di byte o a caratteri (un byte alla volta)

Velocità di trasmissione dei dati



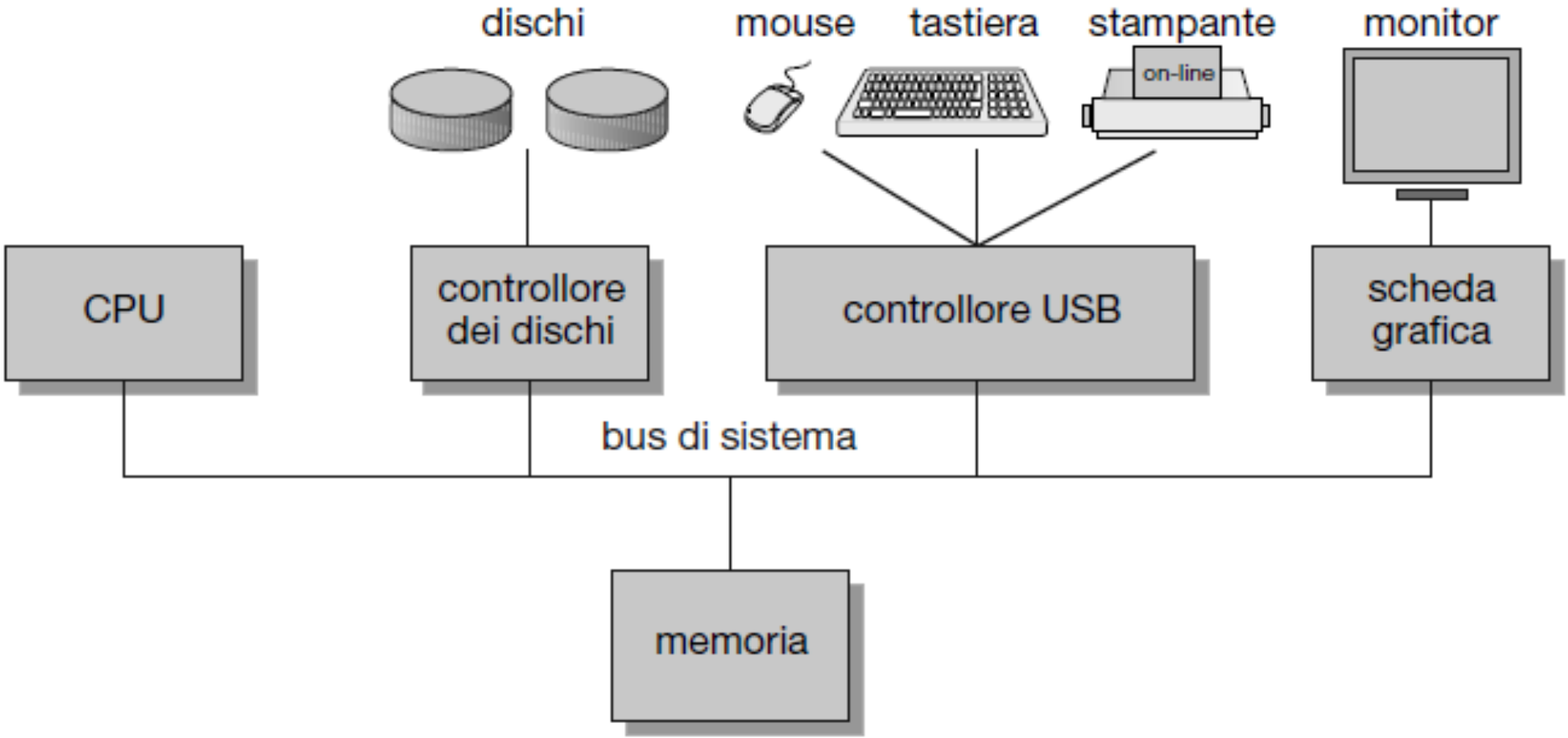
Organizzazione/trasferimento dei dati

- Dispositivi a blocchi (memorie di massa, quali dischi e nastri):
  - registrano i dati in blocchi di dimensione fissa (tipicamente tra 512 byte e 4KB): ciascun blocco (chiamato settore) ha un proprio indirizzo e può essere letto e scritto in maniera indipendente dagli altri
  - i comandi includono *read, write, seek* (se ad accesso casuale)
  - utilizzati di solito tramite un file system (livello d'astrazione più alto)
- Dispositivi a caratteri (tastiere, mouse, stampanti, interfacce di rete):
  - accettano o inviano flussi di caratteri senza poterli strutturare ed indirizzare al loro interno
  - i comandi includono *get* e *put,* non ci sono comandi di posizionamento
  - sopra questa interfaccia solitamente si costruiscono librerie che permettono l'accesso a intere sequenze di caratteri per volta
- Dispositivi speciali: non rientrano nelle due precedenti categorie
  - Timer: temporizzatori programmabili che possono essere regolati in modo da generare interrupt dopo certi intervalli di tempo
  - Touch screen

Hardware di I/O

- Nonostante l'enorme varietà di dispositivi I/O, bastano alcuni concetti per capire come essi sono connessi al sistema e come il SW può controllare l'HW
- L'HW di I/O è costituito da
  - Dispositivi (o periferiche) di ingresso/uscita
  - Controllori dei dispositivi o device controller
  - Componenti di connessione
    - bus (insieme di linee di connessione condivise)
    - porte (punti di connessione)

Schema architetturale semplificato



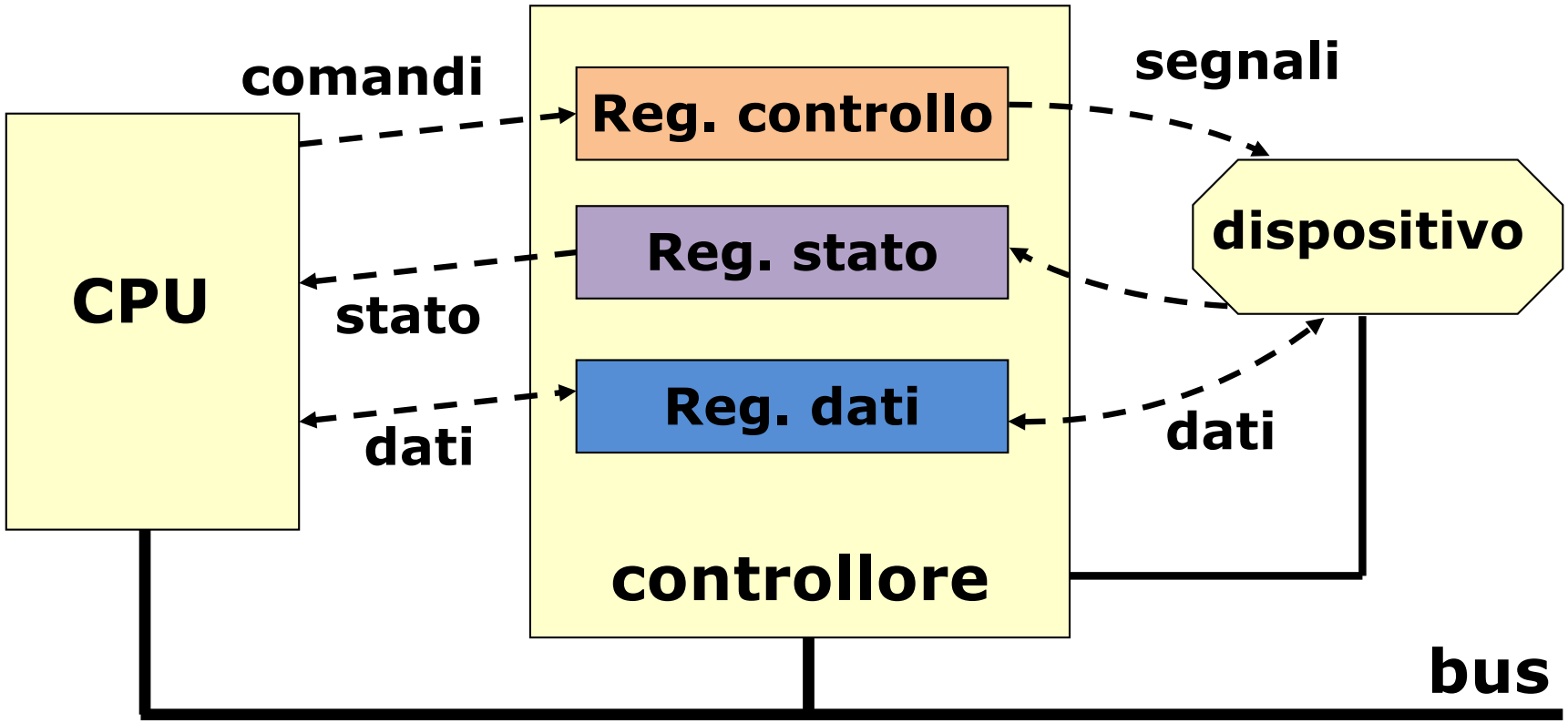
- Controllore: circuito elettronico che collega periferiche e CPU, tramite il bus
- Compiti del controllore
  - controllare il funzionamento del dispositivo inviando e ricevendo segnali dallo stesso tramite un protocollo ben preciso
  - fornire alla CPU un insieme di registri indirizzabili dalle operazioni di I/O

Controllore di un dispositivo di I/O

- Molti controllori possono gestire più dispositivi dello stesso tipo
- L'interfaccia tra controllore e dispositivo è standardizzata (es. SATA, SCSI, USB), cosiché le industrie possono produrre dispositivi e controllori che la usano per interagire
- L'interfaccia è solitamente di livello basso; per esempio,
  - dalla lettura di un settore, un disco produce un flusso seriale di bit
  - il controllore lo struttura in preambolo, byte di dati e codice di correzione degli errori (ECC – error correcting code), quindi
  - prima inserisce i byte di dati in un suo buffer e poi, dopo la correzione di eventuali errori tramite applicazione dell'ECC, copia i byte di dati in memoria principale
- Anche il controllore di un monitor LCD funziona ad un livello basso:
  - legge dalla memoria i caratteri da visualizzare sullo schermo e genera i segnali per modulare la polarizzazione della luce dei pixel corrispondenti
  - se non ci fosse il controllore, spetterebbe al SO programmare i campi elettrici di ciascun pixel!

Controllore di un dispositivo di I/O (schema semplificato)

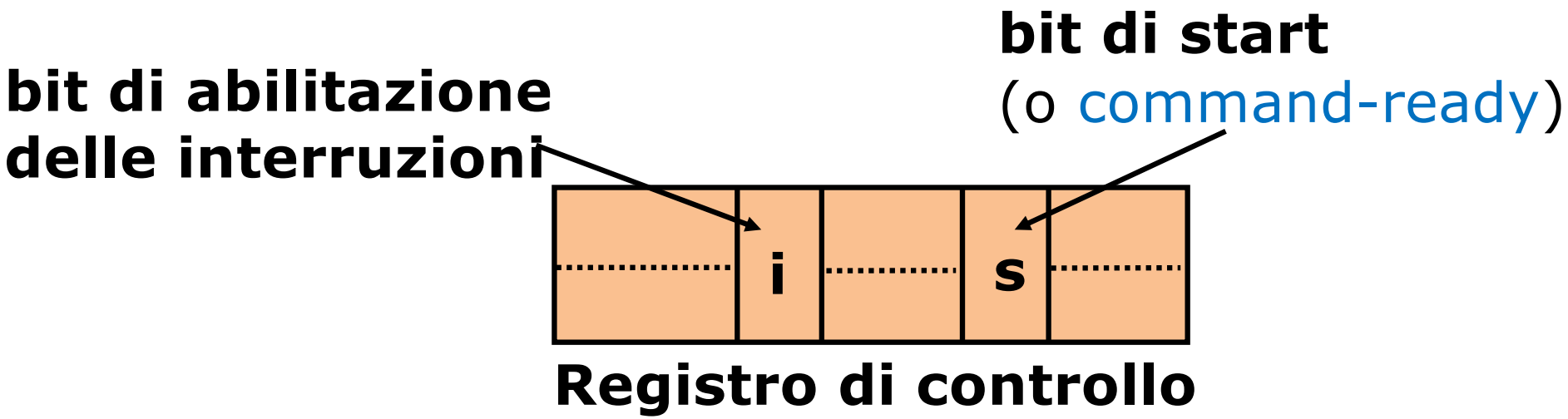
La CPU agisce sul controllore tramite comandi di I/O che indirizzano i registri di cui esso dispone

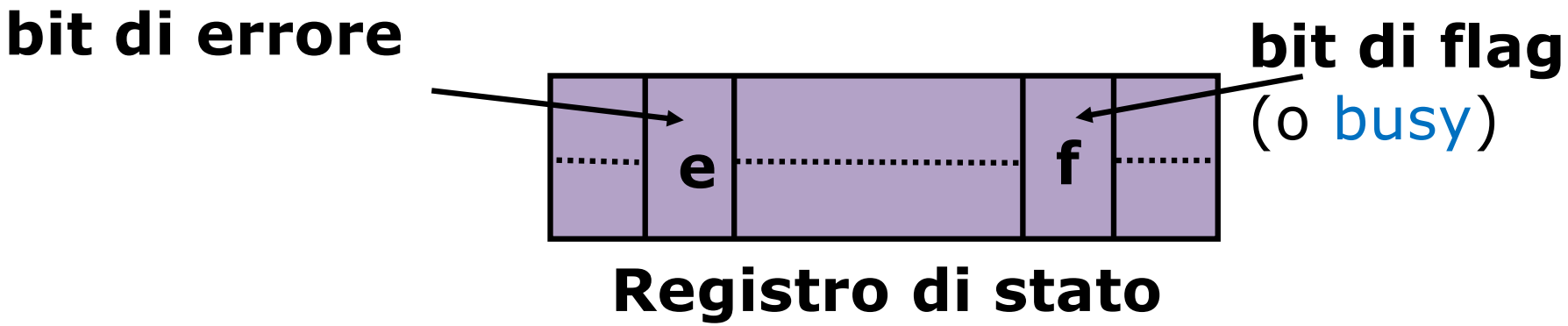


Registri del controllore

- Registro di controllo
  - consente alla CPU di controllare il funzionamento del dispositivo
  - sola scrittura per la CPU che vi inserisce valori opportuni per richiedere al dispositivo lo svolgimento di una operazione
    - **bit di start** = 1: consente al controllore di attivare il dispositivo inviandogli dei segnali
    - **bit di abilitazione delle interruzioni** = 1: consente al controllore di inviare un segnale di interruzione alla CPU alla fine dell'operazione
    - altri bit selezionano l'operazione richiesta
- Registro di stato
  - consente al dispositivo di mantenere aggiornato lo stato in cui si trova
  - sola lettura per la CPU
    - **bit di flag** = 1: alla fine dell'operazione richiesta (alla commutazione da 0 a 1 può eventualmente essere lanciata un'interruzione)
    - **bit di errore** = 1: al verificarsi di un evento anomalo durante il funzionamento del dispositivo (possono esserci tanti bit di errore quante sono le possibili cause di malfunzionamento)
- Registro dati (in/out): buffer del controllore per i dati da trasferire

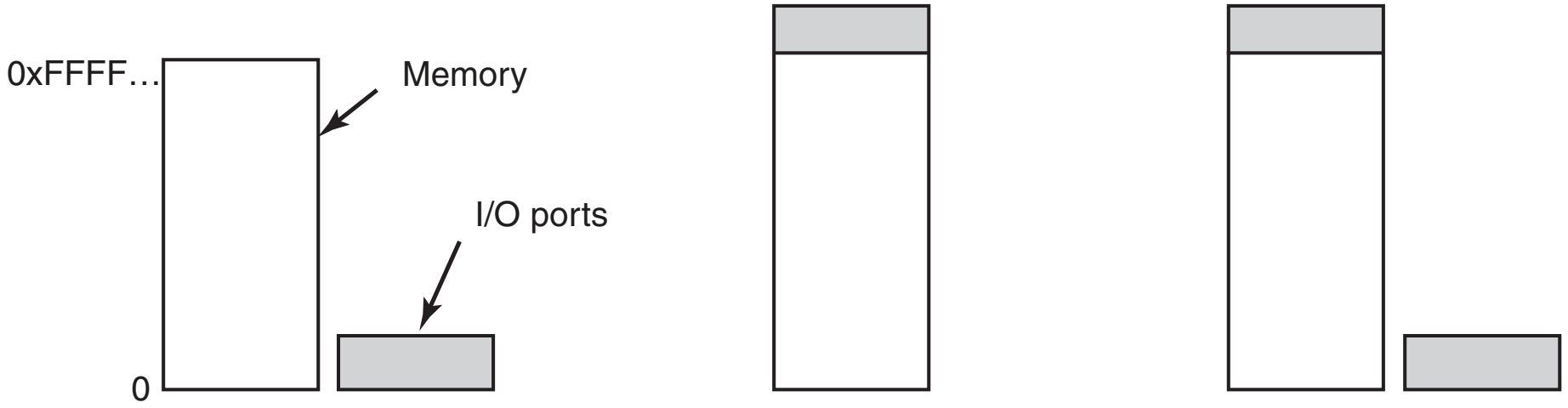
Registri del controllore





Metodi di accesso ai registri di un controllore di dispositivo

Tre metodi diversi per accedere ai registri di un controllore di dispositivo e realizzare la comunicazione tra CPU e controllore



I/O separato in memoria

I/O mappato in memoria

Schema ibrido

I/O separato in memoria (o port-mapped I/O)

- A ciascun registro è assegnato un numero di porta di I/O, es. un intero a 16 bit
- L'insieme di tutte le porte di I/O forma lo spazio delle porte di I/O ed è protetto in modo che i normali programmi utente non possano accedervi
- Può accedere solo il SO tramite apposite istruzioni di I/O che specificano numeri di porte di I/O dedicate ai dispositivi

  `IN R0,4` (la CPU legge il contenuto del registro con porta di I/O `4` e lo inserisce nel registro `R0` interno alla CPU)
- Dato che si utilizzano istruzioni speciali, che si tratti di un I/O è ovvio anche a chi legge un programma scritto in assembly

  `MOV R0,4` (la CPU legge il contenuto della locazione di memoria `4` e lo inserisce nel registro `R0` interno alla CPU) vs. `IN R0,4`
- Vantaggio principale: tutto lo spazio di indirizzi può essere usato per la memoria (utile per CPU con limitate capacità di indirizzamento)
- La maggior parte dei vecchi calcolatori, inclusi tutti i mainframe, come l'IBM360 e i suoi successori, funzionava in questo modo

Indirizzi delle porte di I/O dei dispositivi in un PC (elenco parziale)

indirizzi per l'I/O (in esadecimale)	dispositivo
000-00F	controllore DMA
020-021	controllore delle interruzioni
040-043	timer
200-20F	controllore dei giochi
2F8-2FF	porta seriale (secondaria)
320-32F	controllore del disco
378-37F	porta parallela
3D0-3DF	controllore della grafica
3F0-3F7	controllore dell'unità a dischetti
3F8-3FF	porta seriale (principale)

I/O mappato in memoria (o memory-mapped I/O)

- I registri di controllo dei dispositivi sono mappati in un sottoinsieme dello spazio degli indirizzi di memoria generati dalla CPU
- Ogni controllore di dispositivo monitora il bus degli indirizzi generati dalla CPU e risponde ogni volta che la CPU genera un indirizzo assegnato al dispositivo che controlla
- Efficiente e flessibile
  - Non servono istruzioni speciali per l'I/O: ogni istruzione che può accedere la memoria può anche accedere i registri dei controllori
  - La CPU richiede meno logica interna ed è di conseguenza più economica, veloce e facile da costruire; principio alla base delle architetture RISC (Reduced Instruction Set Computing)
- Accorgimento per la protezione
  - Lo spazio di indirizzamento di I/O è allocato al di fuori dello spazio utente
- Oggigiorno usato dalla maggior parte dei controllori dei dispositivi
  - Introdotto dal PDP-11 (DEC) nei primi anni '70

Schema ibrido

- Con alcuni dispositivi si usa anche uno schema ibrido
- Es: controllore della grafica nel Pentium
  - Registri controllo e stato usano I/O separato in memoria
  - Registro dati usa I/O mappato in memoria
    - Le scritture avvengono tramite la memoria grafica, una regione di memoria dedicata a mantenere i contenuti dello schermo
      - Un thread, per scrivere sullo schermo, inserisce i dati nella memoria grafica
      - Il controllore genera l'immagine dello schermo sulla base del contenuto della memoria grafica
    - Ciò serve a rendere veloce la visualizzazione su schermo (che altrimenti richiederebbe milioni di istruzioni di I/O)

Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

Interazione tra gestore SW e controllore HW

Tre modalità diverse di interazione tra gestore SW di un dispositivo (driver) e relativo controllore HW

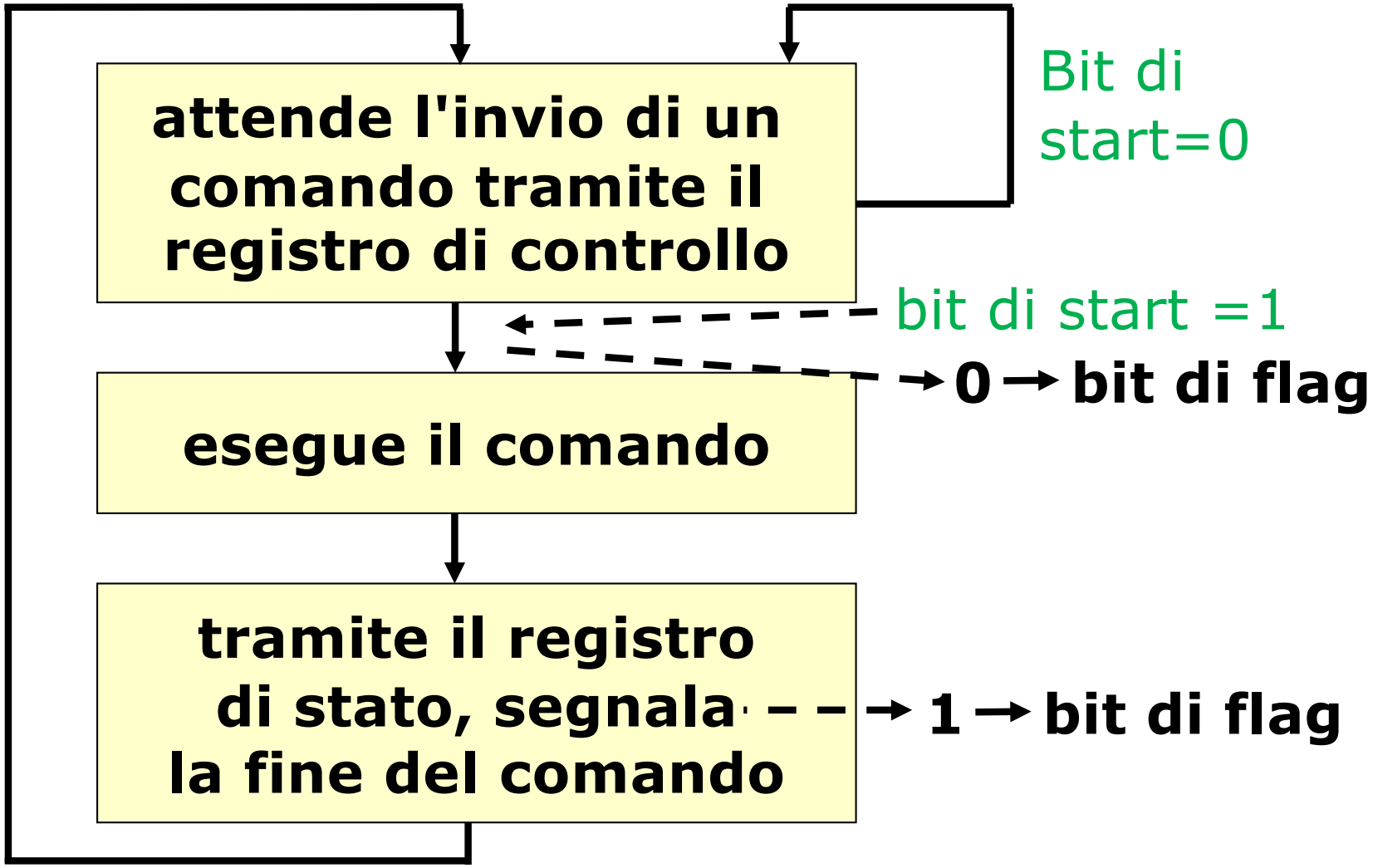
- I/O programmato o polling o a controllo di programma o a controllo diretto o programmed I/O
- I/O guidato dalle interruzioni o interrupt-driven I/O
- Accesso diretto in memoria o direct memory access (DMA)

	Senza interrupt	Con interrupt
Trasferimento tramite CPU	I/O programmato	I/O guidato dalle interruzioni
Trasferimento diretto dispositivo-memoria		DMA (Direct Memory Access)

I/O programmato: processo esterno (HW)

- Un dispositivo con il relativo controllore può essere considerato come un processore dedicato
- Non esegue un programma ma una sequenza di azioni fisse (cablate) che prende il nome di processo esterno (alla CPU)

Processo esterno (HW)



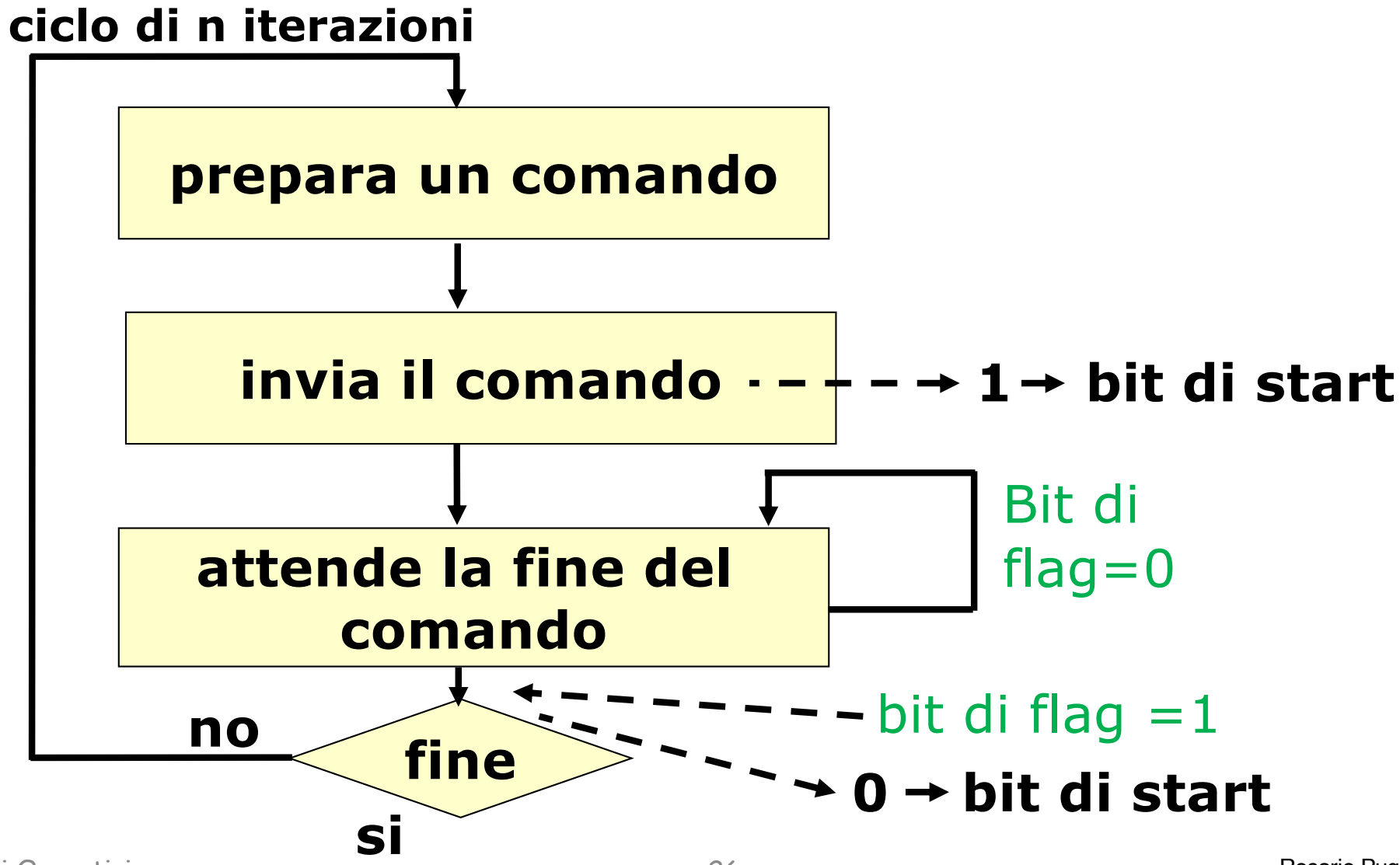
I/O programmato: processo interno (SW)

- Processo che attiva il dispositivo:

  solitamente è il driver (per la gestione) del dispositivo

Processo interno (SW)

L'esecuzione di una istruzione di alto livello per un dispositivo di I/O può corrispondere ad una sequenza di n comandi di basso livello per il controllore



I/O programmato: valutazione

- Tutto il lavoro è delegato alla CPU
  - Il processo interno controlla direttamente la terminazione di ciascuna operazione richiesta al dispositivo, cioè determina lo stato del dispositivo mediante polling (interrogazione ripetuta) del bit di flag
- Problema: attesa attiva (*busy waiting*)
  - Inefficiente, soprattutto se le interrogazioni ripetute trovano raramente un dispositivo pronto per il servizio
  - Ragionevole in sistemi embedded, nei quali la CPU non ha altro da fare
  - Non adatto a sistemi multiprogrammati dove, per utilizzare al meglio la CPU, quando un processo non può proseguire poiché attende un qualche I/O, è necessario sospenderlo commutando la CPU

I/O guidato dalle interruzioni

- È possibile rendere più efficiente la gestione associando al dispositivo un semaforo di sospensione `S` (inizializzato a 0)
- Nel processo interno, il ciclo di attesa attiva sulla condizione che il contenuto del registro di stato deve soddisfare è sostituito da una **`wait`**`(S)` sul semaforo (con conseguente, eventuale, sospensione del processo)
- Al completamento dell'operazione da parte del dispositivo, il controllore del dispositivo lancia un segnale di interruzione
- La CPU controlla la presenza di segnali di interruzione dopo l'esecuzione di ogni istruzione; quando ne rileva uno, salva lo stato corrente e passa all'esecuzione della routine di gestione
- La routine attiva l'esecuzione della funzione di risposta alle interruzioni da dispositivo la quale, tra le altre attività, esegue una **`signal`**`(S)` che provoca lo sblocco del processo interno

I/O guidato dalle interruzioni

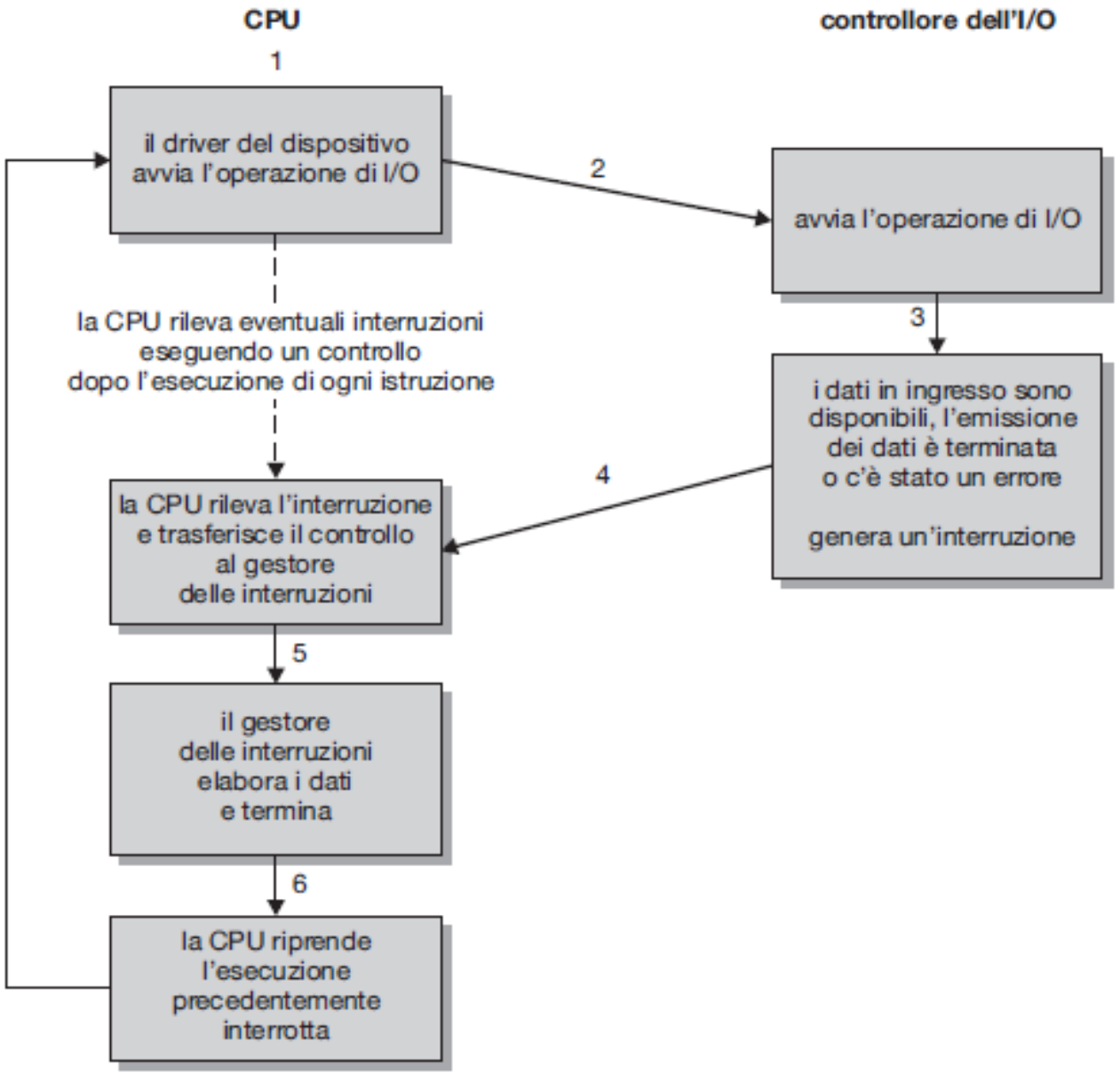
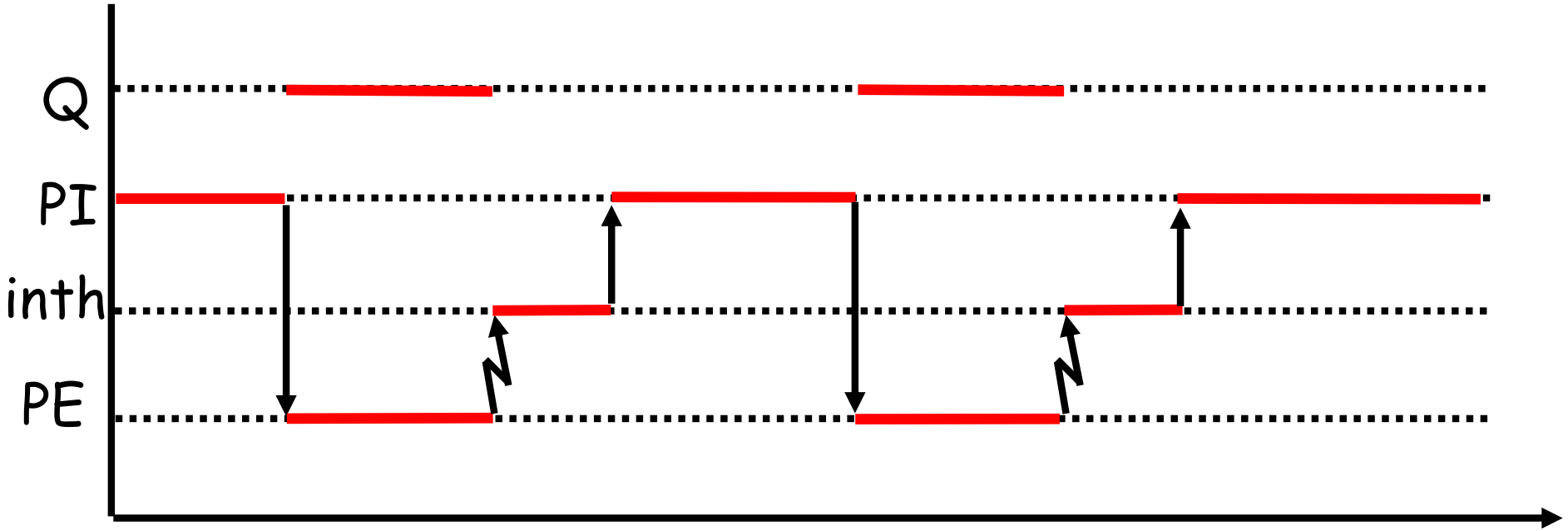


Diagramma temporale della gestione a interruzione (migliorata)

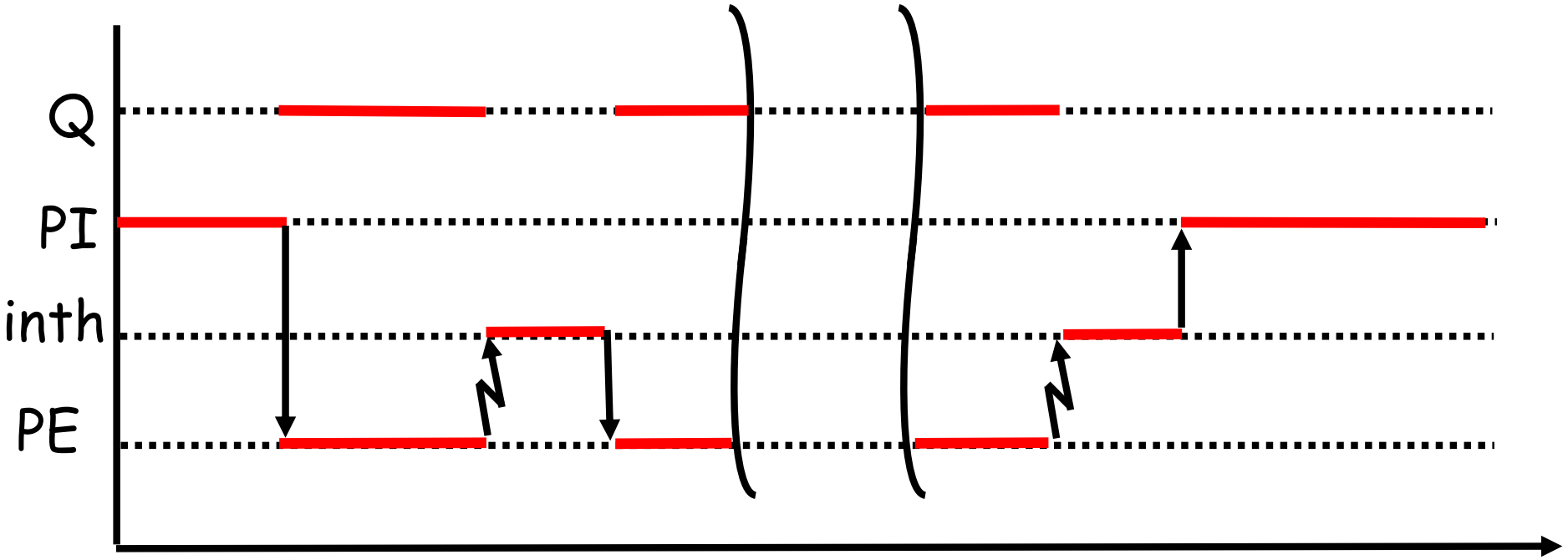


PI: processo interno (attiva il dispositivo)
PE: processo esterno
inth: routine di gestione interruzioni
Q: processo applicativo

Miglioramento

- Inconveniente: se il processo interno deve trasferire n blocchi di dati, per n-1 volte viene sospeso e riattivato inutilmente in quanto deve subito (ri)sospendersi
- Se è disponibile una funzione che accetta come argomento il numero n dei blocchi di dati da trasferire e l'indirizzo di memoria dove memorizzarli o da dove leggerli, si può bloccare il processo interno fino al trasferimento dell'ultimo blocco e riattivarlo solo allora

Diagramma temporale della gestione a interruzione (migliorata)



PI: processo interno (attiva il dispositivo)
PE: processo esterno
inth: routine di gestione interruzioni
Q: processo applicativo

Miglioramento

- Inconveniente: se il processo interno deve trasferire n blocchi di dati, per n-1 volte viene sospeso e riattivato inutilmente in quanto deve subito (ri)sospendersi
- Se è disponibile una funzione che accetta come argomento il numero n dei blocchi di dati da trasferire e l'indirizzo di memoria dove memorizzarli o da dove leggerli, si può bloccare il processo interno fino al trasferimento dell'ultimo blocco e riattivarlo solo allora
- Per ottenere questo comportamento la routine di gestione dell'interruzione deve poter distinguere le interruzioni intermedie da quella finale in corrispondenza della quale il processo interno va riattivato
- Ciò è realizzabile grazie al descrittore di dispositivo

Descrittore di un dispositivo

- Struttura dati in memoria che rappresenta il dispositivo ed a cui possono accedere sia il processo interno che la routine di gestione dell'interruzione
- Duplice scopo
  - racchiude al suo interno le informazioni associate al dispositivo
  - consente la comunicazione tra processo interno e controllore del dispositivo (quantità di dati da trasferire, indirizzo del buffer in/da cui trasferirli) e viceversa (esito del trasferimento)

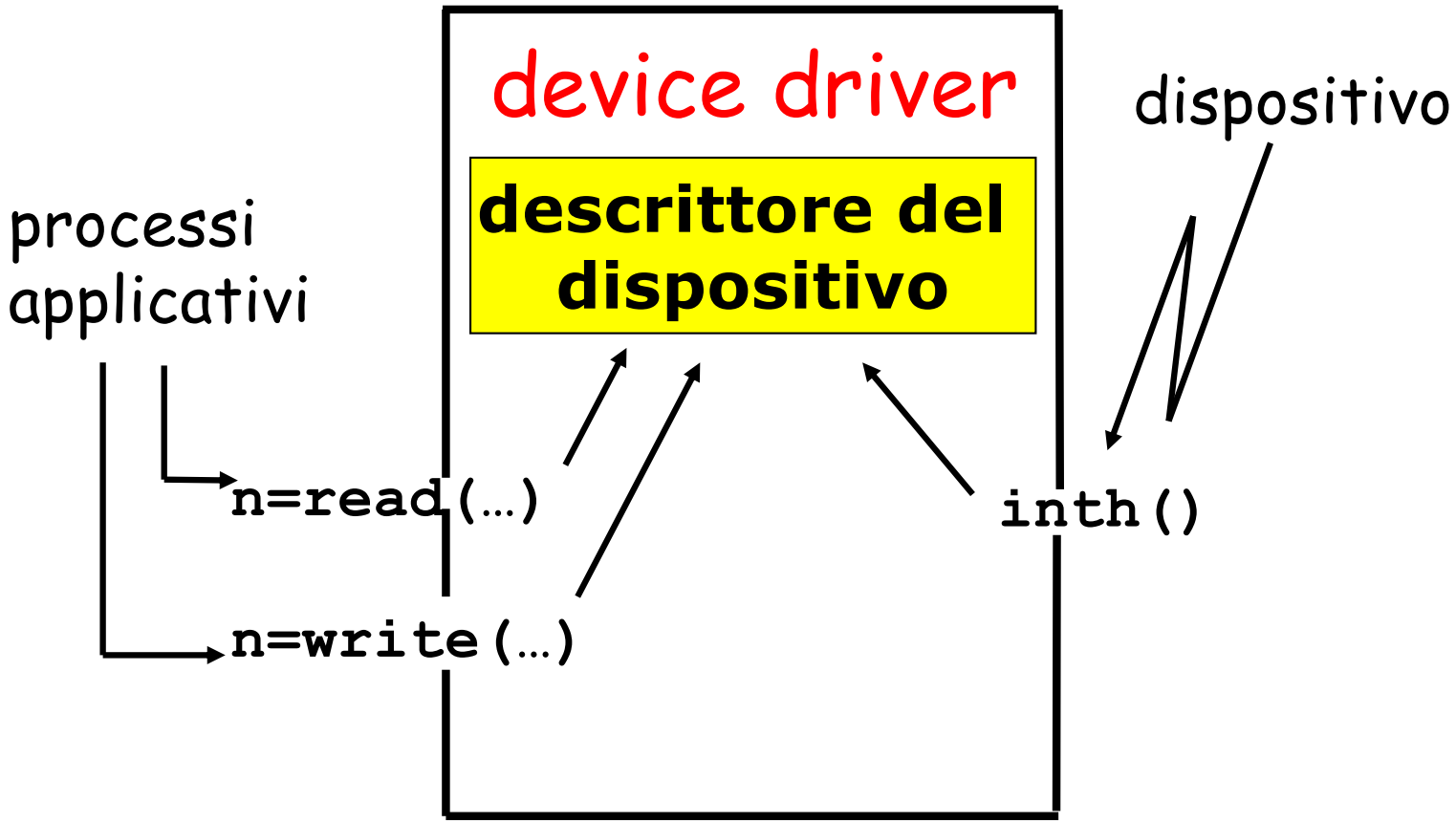
Campi del descrittore di un dispositivo

indirizzo registro di controllo
indirizzo registro di stato
indirizzi registri dati
semaforo `dato_disponibile`
quantità di dati da trasferire `contatore`
indirizzo del buffer in memoria `puntatore`
esito del trasferimento `esito`

Gestore di un dispositivo (*device driver*)

Componente SW formata da

- descrittore del dispositivo
- funzioni di accesso al dispositivo (lettura, scrittura, ...)
- funzioni di risposta alle interruzioni



Gestore di un dispositivo: funzionamento tipico

1. Inizializza il dispositivo
2. Accetta richieste di operazione e ne controlla la correttezza
3. Gestisce le code delle richieste che non possono essere servite subito
4. Sceglie la prossima richiesta da servire e la traduce in una sequenza *S* di comandi a basso livello da inviare al controllore del dispositivo
5. Trasmette al controllore i comandi in *S*, uno alla volta, eventualmente bloccandosi in attesa del completamento dell'esecuzione del comando
6. Controlla l'esito di ciascun comando gestendo eventuali errori
7. Al completamento della richiesta, invia l'esito della richiesta ed eventuali dati al processo richiedente

Gestione di un timer

- Utilizzato per lanciare interruzioni cadenzate nel tempo (non per trasferire dati)
  - Gestione della data (orologio di sistema)
  - Scheduling della CPU nei sistemi time-sharing
  - Attese programmate e ricezione di segnali di time-out
    - Il sottosistema di I/O lo usa per riversare periodicamente sui dischi il contenuto del buffer cache
    - Il sottosistema di rete lo usa per annullare operazioni che hanno una latenza eccessiva
- Il driver è costituito da
  - **`delay()`**: primitiva che i processi possono invocare passandogli come argomento un intero che indica la durata dell'attesa richiesta
  - Descrittore: oltre ai campi relativi agli indirizzi dei registri di controllo e di stato, ha i seguenti campi
    - **`fine_attesa[]`**: array di semafori su cui si sospendono i processi che invocano **`delay()`**
    - **`ritardo[]`**: array di interi che indicano le unità di tempo che devono ancora trascorrere prima che ogni singolo processo possa essere risvegliato
    - contatore: un registro in cui la CPU può scrivere un intero; il timer lo decrementa periodicamente, quando si azzera lancia un'interruzione e reimposta contatore con il valore più piccolo contenuto nell'array **`ritardo[]`**

Descrittore di un timer

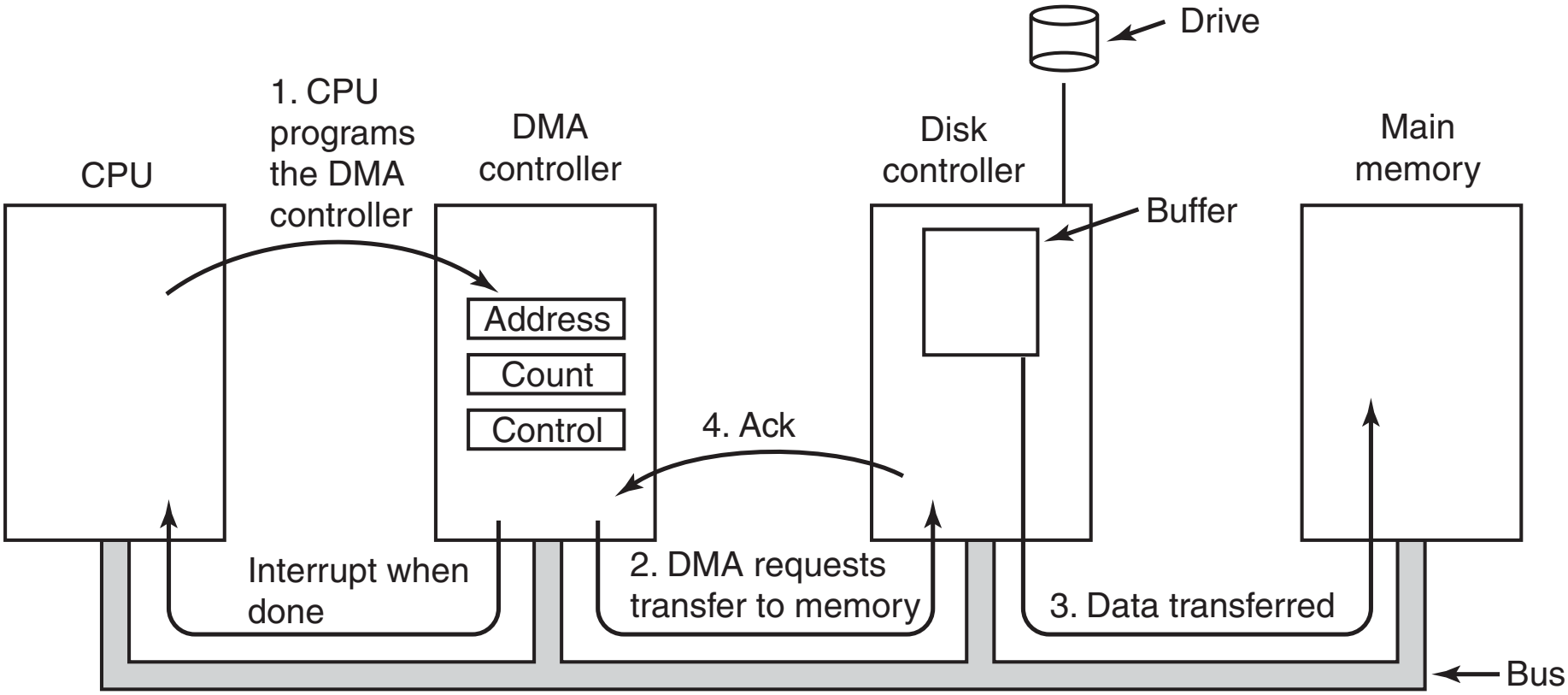
Gestione di un dispositivo in DMA

- Si usa per trasferimenti di grandi quantità di dati ma richiede HW apposito: un controllore (o canale) DMA
- Consente di trasferire i dati dal registro dati del controllore del dispositivo direttamente alla memoria centrale (e viceversa) tramite lo stesso bus usato dalla CPU per i trasferimenti con la memoria, ma senza intervento della CPU
- Il controllore DMA può essere integrato nel controllore di un dispositivo (in questo caso ne serve uno per ogni dispositivo) oppure separato (a volte è sulla scheda madre)
- Nel controllore DMA sono presenti 2 registri, puntatore e contatore, con significato simile ai corrispondenti campi del descrittore di dispositivo ma gestiti via HW
  - Nel descrittore del dispositivo tali campi mancano
- Il controllore DMA ha inoltre uno o più registri di controllo in cui la CPU inserisce i comandi appropriati

Lettura in DMA da un disco (la scrittura è simile)

- La CPU programma il controllore DMA impostando anche i registri *puntatore* e *contatore* e gli delega il compito di gestire il trasferimento in memoria tramite il bus
- La CPU quindi programma il controllore del disco per la lettura dei dati
- Quando nel suo buffer ci sono dati validi, e pronti per il trasferimento, il controllore del disco invia un segnale al controllore DMA
- Quando il controllore DMA intercetta il segnale inizia il trasferimento inviando sul bus una richiesta rivolta al controllore del disco con la specifica di un indirizzo di memoria
- Quando il trasferimento è stato eseguito, il controllore del disco invia un segnale di conferma al controllore DMA
- Il controllore DMA decrementa il registro *contatore* e incrementa il registro *puntatore* e invia un interrupt alla CPU oppure rilancia una nuova richiesta per il controllore del disco a seconda che il valore del registro contatore si sia azzerato o no

Operazioni durante un trasferimento DMA



Controllore DMA separato dal controllore del disco

Gestione di un dispositivo in DMA

- Vantaggi
  - Riduce il tempo per trasferire un insieme di blocchi di dati
  - Elimina la necessità da parte della CPU di eseguire la funzione di gestione delle interruzioni per ciascun blocco letto (genera una sola interruzione, alla fine dell'intero trasferimento)
- Inconvenienti: il controllore DMA è in generale molto più lento della CPU
  - se il controllore DMA non è in grado di condurre un dispositivo a velocità massima, oppure
  - se la CPU non ha altro da fare mentre attende l'interrupt generato per segnalare la fine del trasferimento
  - allora, per motivi economici (aspetto non trascurabile, per esempio nei sistemi embedded), può valer la pena usare I/O guidato dagli interrupt o I/O programmato

Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

Sottosistema di I/O

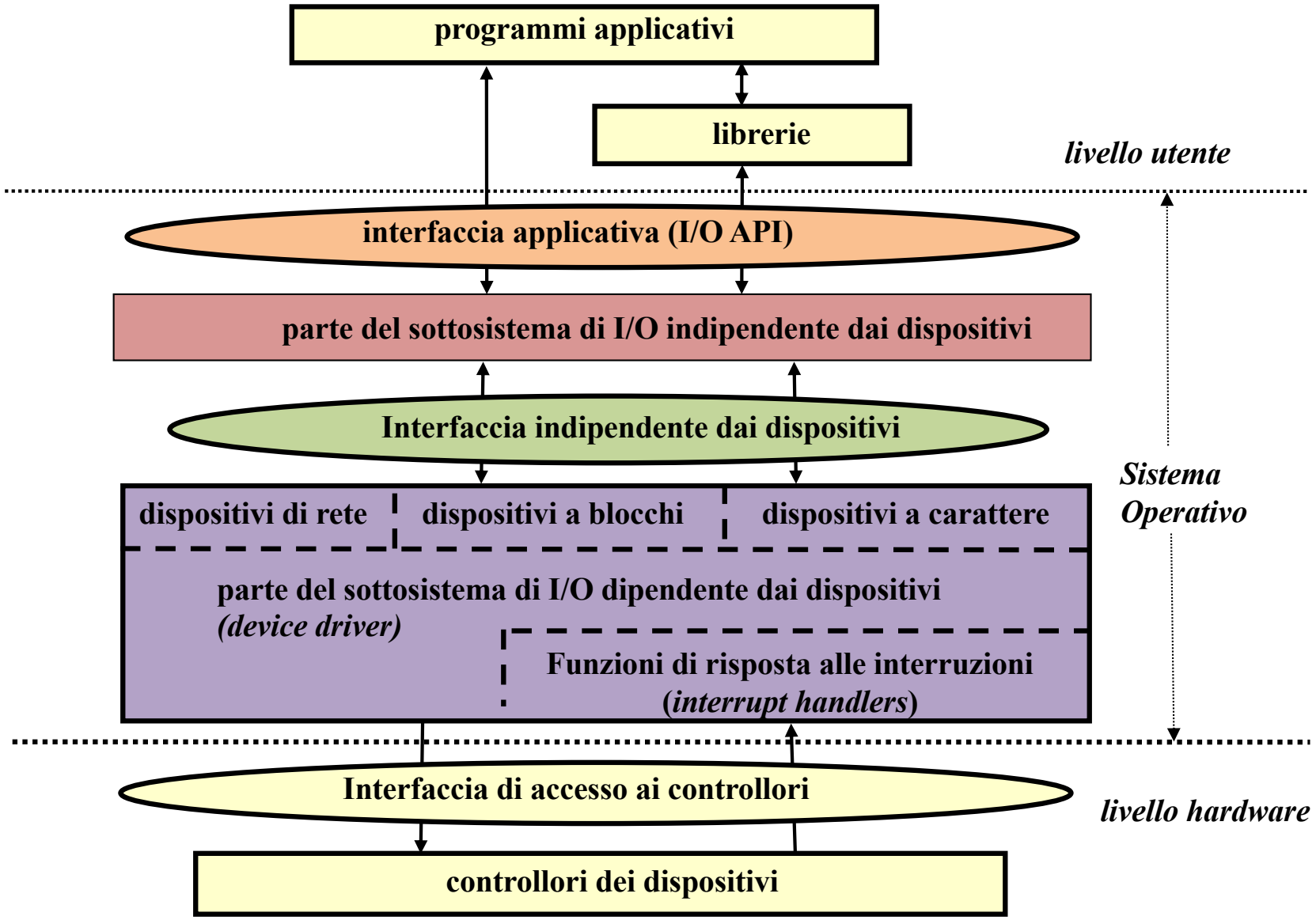
- È formato dai metodi utilizzati dal kernel del SO per la gestione dei dispositivi di I/O
- Separa il resto del kernel dalla complessità di gestione dei dispositivi di I/O dovuta alla loro grande varietà
- Offre un insieme di servizi per assicurare ai processi un supporto adeguato all'esecuzione delle operazioni di ingresso/uscita
- Fornisce un'interfaccia uniforme che
  - garantisce un accesso efficiente ai dispositivi, gestendo anche eventuali malfunzionamenti
  - nasconde i dettagli HW dei singoli (controllori dei) dispositivi
    - Es. evita che un programma che deve leggere alcuni record di un file debba cambiare sequenza di comandi a seconda del dispositivo su cui il file risiede (es. disco fisso, disco a stato solido, DVD, penna USB)

Organizzazione logica del sottosistema di I/O

Il sottosistema di I/O è logicamente suddiviso in 2 componenti

- Componente indipendente dai dispositivi
  - Omogeneizza le funzioni di accesso ai vari dispositivi e fornisce un'interfaccia uniforme al SW a livello utente
  - Fornisce un'Interfaccia di Programmazione Applicativa per l'ingresso/uscita (I/O API): un insieme di funzioni che incapsulano il comportamento delle periferiche in poche classi generiche e invocano le funzioni fornite dai driver dei dispositivi
- Componente dipendente dai dispositivi
  - Nasconde le peculiarità dei dispositivi e dei loro controllori al resto del SO
  - È costituita dai gestori dei dispositivi (*device driver*) che si interfacciano direttamente con i corrispondenti controllori HW, tramite i loro registri, e ne controllano il comportamento

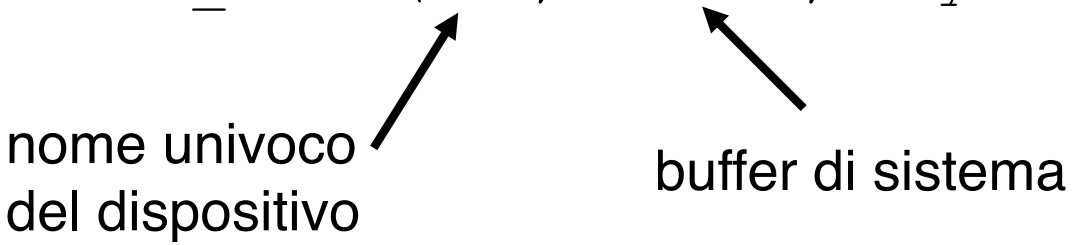
Organizzazione logica del sottosistema di I/O



Componente dipendente dai dispositivi

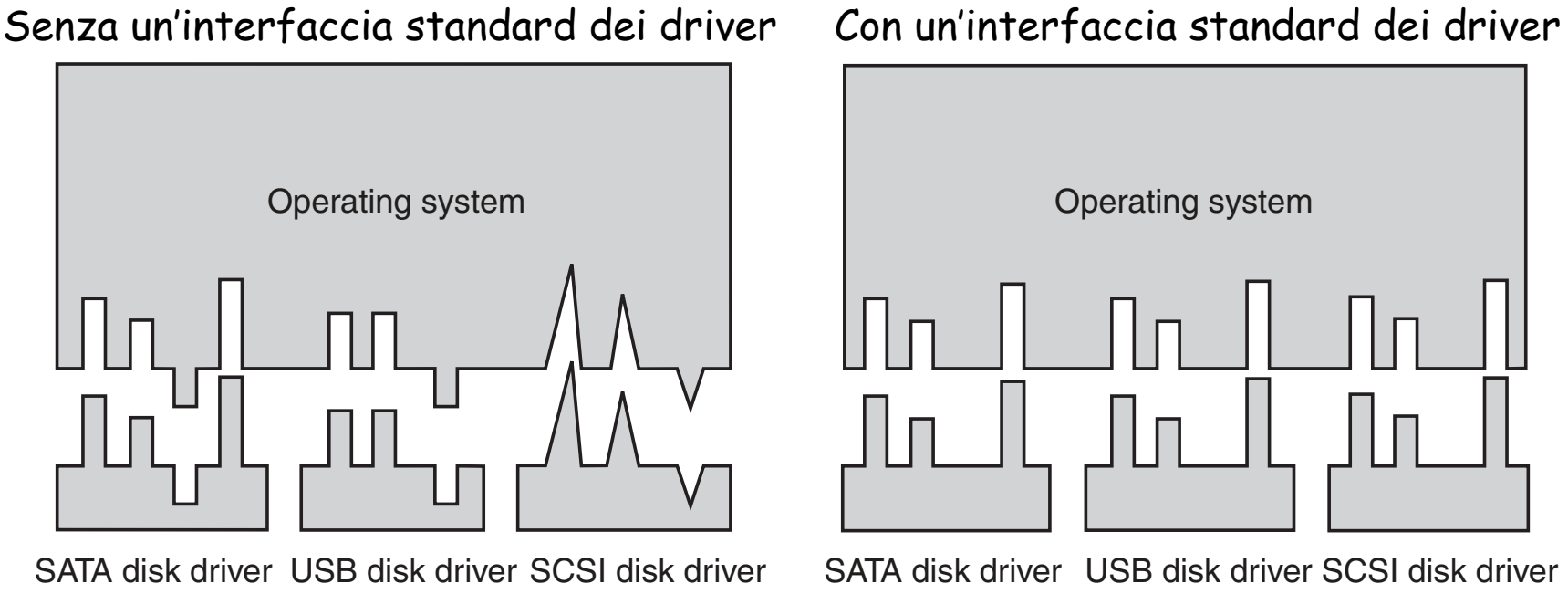
Costituita dai gestori dei dispositivi (device driver) che offrono al livello superiore un insieme di funzioni di accesso ai dispositivi

```
n = _read (dev, buffer, nbytes)
```



Interfacciamento uniforme dei driver

- Per ciascuna classe di dispositivi, come dischi e stampanti, il SO stabilisce un insieme di funzioni che il driver deve supportare
- Così facendo è semplice aggiungere nuovi dispositivi al sistema



Componente indipendente dai dispositivi

- Offre servizi che rendono più semplice, sicuro ed efficiente l'uso dei dispositivi e del calcolatore
- In alcuni sistemi, i servizi offerti a questo livello coincidono con quelli offerti dal file system (essendo uniformata a livello di API la gestione dei file e quella dei dispositivi periferici)
- Principali servizi realizzati a questo livello:
  - buffering dei dati
  - caching dei dati
  - gestione degli errori
  - protezione dell'I/O
  - prenotazione dei dispositivi
  - sistema di spooling
  - scheduling delle richieste di I/O

Buffering dei dati

- È la memorizzazione temporanea, e trasparente al richiedente, dei dati soggetti al trasferimento in buffer di sistema
- Un buffer (memoria tampone) è un'area di memoria che memorizza i dati durante il trasferimento tra due dispositivi o tra un dispositivo e un'applicazione
- Si ricorre ai buffer principalmente per tre ragioni
  - Gestire la differenza di velocità tra produttore e consumatore di un flusso di dati disaccoppiandone il funzionamento (es. modem vs. disco, dispositivi a carattere vs. CPU)
  - Gestire dispositivi che trasferiscono dati in blocchi di dimensioni diverse (disparità comuni, per esempio, nelle reti di calcolatori dove i buffer sono spesso utilizzati per frammentare e riassemblare messaggi)
  - Supportare la 'semantica della copia' nell'ambito delle operazioni di I/O delle applicazioni (garantire che i dati copiati su disco siano quelli contenuti nel buffer al momento dell'invocazione della system call `write()`, indipendentemente da ogni modifica successiva, anche in caso di scrittura asincrona)

Caching dei dati

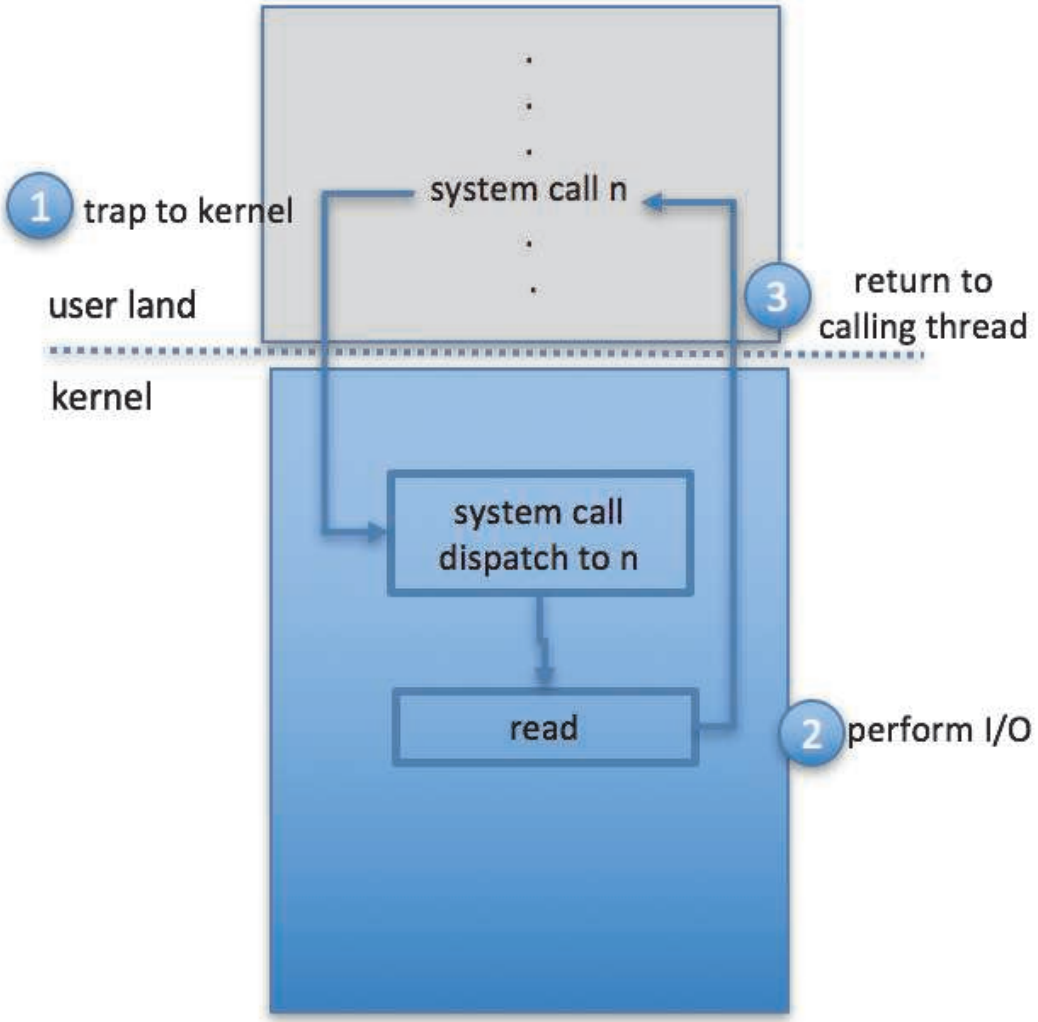
- Una cache è una regione di memoria veloce che serve per mantenere copie di dati: l'accesso alla copia dei dati nella cache è più rapido dell'accesso ai dati originali
  - Ad esempio, le istruzioni del processo attualmente in esecuzione sono memorizzate su disco, copiate nella memoria principale che funge da cache del disco e copiate ulteriormente nella cache primaria e secondaria della CPU
- La differenza tra un buffer e una cache è che un buffer può contenere dati di cui non esiste alcun altra copia, mentre una cache, per definizione, conserva in una memoria più veloce una copia di dati memorizzati altrove
- La memorizzazione nella cache e il buffering sono funzioni distinte, ma a volte una stessa regione di memoria può essere utilizzata per entrambi gli scopi
  - Es. il buffer cache di cui abbiamo parlato nella gestione della memoria secondaria

Gestione degli errori

- I dispositivi e i trasferimenti di I/O possono fallire in molti modi, sia per ragioni contingenti, es. sovraccarico della rete, sia per motivi permanenti, es. un controllore del disco che diventa difettoso
- I SO sono spesso capaci di compensare efficacemente le conseguenze negative dei guasti temporanei; ad esempio,
  - se una system call `read()` non ha successo, il sistema ritenta la lettura
  - se una system call `send()` provoca un errore, il protocollo di rete può richiedere una `resend()`
- Invece, difficilmente i SO riescono a compensare gli effetti di errori dovuti a guasti permanenti di una componente importante
- In generale, una system call di I/O restituisce un bit di informazioni sullo stato d'esecuzione della chiamata, indicando la riuscita o l'insuccesso dell'operazione richiesta
- In UNIX, viene utilizzata una variabile intera aggiuntiva denominata `errno` per restituire un codice di errore (uno tra un centinaio di valori possibili) che indica piuttosto genericamente l'errore occorso (es. puntatore non valido, file non aperto, argomento oltre i limiti)
- Al contrario, alcuni dispositivi possono fornire informazioni di errore molto dettagliate, sebbene molti SO attuali non siano progettati per passare queste informazioni alle applicazioni

Protezione dell'I/O

- Le istruzioni di I/O sono privilegiate, così da impedire agli utenti di eseguirle direttamente
- Per eseguire operazioni di I/O, un programma utente invoca una system call affinché il SO esegua l'I/O per suo conto
- Il SO, lavorando in modalità kernel, verifica che la richiesta sia valida e, se lo è, esegue l'I/O richiesto e, alla fine, restituisce il controllo all'utente in modalità utente



Prenotazione dei dispositivi

- Alcuni dispositivi, come unità nastro e stampanti, che non possono accettare flussi di dati intercalati provenienti da richieste concorrenti, possono essere allocati dinamicamente ad un solo processo per volta
- Alcuni SO (es. VMS) consentono ai processi di richiedere l'uso esclusivo di un dispositivo e di rilasciarlo dopo l'uso
  - Richieste/rilasci possono essere effettuate tramite le system call `open/close` sul file speciale corrispondente al dispositivo, col vincolo che si può fare una sola `open` per volta
  - Il SO esamina le richieste per l'uso del dispositivo e le accetta o le rifiuta, a seconda che il dispositivo sia disponibile o meno
    - Le richieste non accettate provocano il blocco del processo richiedente ed il suo inserimento in una coda di attesa per il dispositivo
    - I rilasci provocano l'assegnazione del dispositivo al processo in coda con maggiori diritti di usarlo e lo sblocco del processo stesso
- Altri SO forniscono funzioni che consentono ai processi di coordinare l'accesso esclusivo tra loro
  - Es. Windows fornisce system call per consentire ad un'applicazione di attendere fino a quando un certo dispositivo diventa disponibile
  - Su questi sistemi, spetta alle applicazioni evitare il deadlock

Sistema di spooling

- Lo spooling (simultaneous peripheral operation on-line) è una tecnica alternativa all'allocazione dinamica tramite richieste e rilasci per gestire dispositivi che non possono accettare flussi di dati intercalati
  - Es. sebbene una stampante possa servire una sola richiesta alla volta, diverse applicazioni devono poter richiedere simultaneamente la stampa dei propri risultati, senza che le stampe si mischino
- Il SO risolve questo problema
  - intercettando i dati provenienti da ogni singola applicazione e indirizzati alla stampante
  - memorizzando tali dati in uno specifico file su disco (chiamato file di spool)
  - inserendo (quando l'applicazione termina di produrre i dati da stampare) il file di spool nella coda stampa
  - copiando, uno alla volta, i file dalla coda di stampa alla stampante
- Si crea quindi una coda di file in attesa di essere elaborati anziché una lista di richieste pendenti per l'assegnazione della risorsa
- In alcuni SO, lo spooling è gestito da un processo daemon di sistema, in altri è gestito da un thread nel kernel
- In entrambi i casi, il SO fornisce un'interfaccia di controllo che permette ad utenti ed amministratori di esaminare la coda di stampa, eliminare elementi prima che vengano stampati, sospendere il servizio di stampa, …

Scheduling delle richieste di I/O

- Consiste nel determinare un ordine conveniente con cui servire le richieste di I/O in una coda
  - L'ordine in cui tali richieste sono generate dalle applicazioni raramente è la scelta migliore
- Viene realizzato mantenendo una coda di richieste per ciascun dispositivo
  - Quando un'applicazione richiede un'operazione di I/O si inserisce la richiesta nella coda del relativo dispositivo
  - In caso di I/O asincrono, per tener traccia delle richieste di I/O attive ad un dato istante si usa una tabella di stato dei dispositivi che contiene una voce per ogni dispositivo di I/O che ne indica tipo, stato (in attesa, occupato) e coda delle richieste pendenti
  - Lo scheduler dell'I/O riorganizza l'ordine delle richieste in coda per
    - migliorare le prestazioni complessive del sistema
    - distribuire equamente gli accessi dei processi ai dispositivi
    - ridurre il tempo di attesa medio per il completamento di un'operazione di I/O
- Esamineremo in maniera specifica diversi algoritmi di scheduling delle richieste di lettura/scrittura su disco

SW di I/O nello spazio utente

- La maggior parte del SW di I/O è nel SO, una piccola parte è a livello utente
- Il SW di I/O nello spazio utente consiste sostanzialmente delle funzioni di libreria che invocano le I/O API e che sono a loro volta invocate dai (e collegate ai) programmi applicativi
  - Alcune funzioni non fanno altro che inserire i parametri con cui sono invocate nel posto opportuno per invocare la system call corrispondente

```
count = write(fd, buffer, nbytes);
```

  - Altre svolgono a tutti gli effetti il lavoro reale, come ad esempio la formattazione dell'I/O effettuata da `printf`

```
printf("The square of %3d is %6d\n",i,i*i);
```

Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

Traduzione dei nomi

- Permette di associare ai nomi simbolici dei file usati dalle applicazioni i dispositivi hardware corrispondenti
- Si tratta di un processo che passa attraverso diversi stadi
  - dalla stringa di caratteri che rappresenta il nome logico
  - a un driver specifico e all'indirizzo di un dispositivo
  - fino all'indirizzo fisico delle porte di I/O o all'indirizzo mappato in memoria del controllore del dispositivo

Traduzione dei nomi

- Il file system ha il compito di fornire il modo di giungere, attraverso la struttura delle directory, alla regione del disco dove i dati del file sono fisicamente residenti
- Ad esempio
  - Nel file system FAT, il nome del file è associato ad un numero che indica una voce nella tabella di allocazione dei file e quella voce della tabella, a sua volta, permette di individuare i blocchi del disco in cui è allocato il file
  - In UNIX, il nome del file è associato ad un numero di inode il quale contiene le informazioni necessarie per individuare i blocchi del disco in cui è allocato il file

Traduzione dei nomi

- Ma come viene effettuato il collegamento tra il nome del file e il controllore del disco (indirizzo fisico della porta di I/O o registri mappati in memoria del controllore)?
- Questo collegamento può avvenire
  - in uno specifico spazio di nomi dei dispositivi (FAT)
  - all'interno dello spazio di nomi del file system (UNIX)

Traduzione dei nomi: FAT

- Lo spazio dei nomi dei dispositivi è distinto dallo spazio dei nomi del file system grazie all'uso dei due punti come separatore
- La prima parte di un nome di file, quella che precede i due punti, è una stringa che identifica un dispositivo HW specifico
  - Ad esempio, C: è la prima parte di ogni nome di file che risiede sul disco rigido principale
  - La convenzione che C rappresenti il disco rigido principale è incorporata nel SO
  - C è associato a uno specifico indirizzo di porta per mezzo di una tabella dei dispositivi

Traduzione dei nomi: UNIX

- In un nome di file (pathname) UNIX, il nome del dispositivo non è presente esplicitamente (diversamente da FAT)
- UNIX invece usa una tabella di montaggio che associa prefissi di pathname ai nomi dei dispositivi corrispondenti
  - Per risolvere un pathname, UNIX esamina la tabella di montaggio per trovare il più lungo prefisso corrispondente
  - La voce relativa a quel prefisso nella tabella di montaggio fornisce il nome del dispositivo voluto
- Nella struttura della directory del file system, un nome di dispositivo è associato, non al numero di un inode, ma ad una coppia di numeri <principale, secondario> che identifica il dispositivo
  - Il numero principale identifica il driver di dispositivo che deve essere invocato per gestire l'I/O sul dispositivo in questione
  - Il numero secondario deve invece essere passato come parametro a questo driver affinché esso possa determinare, per mezzo della tabella dei dispositivi, l'indirizzo fisico della porta di I/O o l'indirizzo mappato in memoria del controllore del dispositivo interessato

Traduzione dei nomi

- I SO moderni ottengono una notevole flessibilità grazie all'uso di diverse tabelle di ricerca a vari livelli durante il processo di traduzione
  - Pertanto è possibile introdurre nuovi dispositivi e driver in un computer senza dover ricompilare il kernel
- In effetti, alcuni SO sono in grado di caricare i driver dei dispositivi su richiesta
- Al momento dell'avvio,
  - il sistema controlla innanzitutto i bus HW per determinare quali dispositivi sono presenti
  - carica quindi i driver necessari, immediatamente o quando servono per soddisfare una richiesta di I/O
- Dopo l'avvio, i dispositivi aggiunti
  - possono essere rilevati grazie all'errore che causano (dovuto alla generazione di un interrupt che non ha un gestore di interrupt associato)
  - il quale può richiedere al kernel di ispezionare i dettagli del dispositivo e caricare dinamicamente un driver di dispositivo appropriato

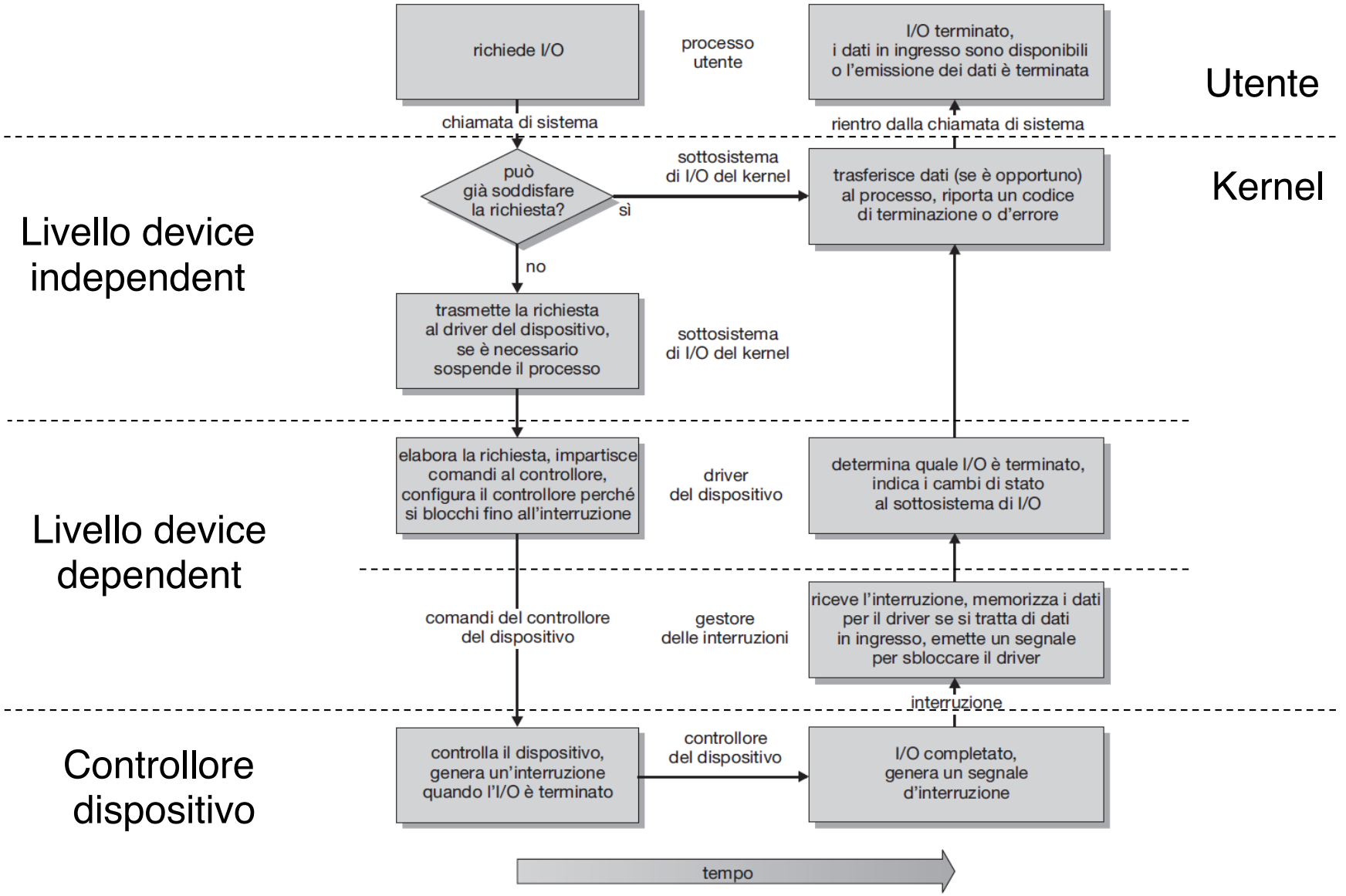
Passi per l'esecuzione di una `read(fd,bytes,&buffer)`

- Un processo invoca una system call `read()` bloccante su un file descriptor `fd` restituito da una precedente `open()`
- Il codice della system call nel kernel controlla la correttezza dei parametri
- Il livello "File system logico" del file system blocca l'operazione di lettura se non è permessa sulla base dei diritti di accesso del processo relativamente al file
- Se i dati sono già disponibili nel buffer cache, vengono restituiti al processo invocante e la system call termina
- Altrimenti, è necessario eseguire un'operazione di I/O dal disco
  - il processo richiedente viene posto in stato waiting e il suo PCB è rimosso dalla coda dei processi pronti e inserito nella coda di attesa sul dispositivo
  - il livello "Modulo di organizzazione dei file" del file system calcola il record logico (o i record logici) del file da leggere e il blocco del disco corrispondente utilizzando le informazioni nella tabella dei file aperti
  - il livello "File system di base" invia quindi una richiesta di lettura al driver del dispositivo (o tramite chiamata di procedura o tramite messaggio interno al kernel, a seconda del SO)

Passi per l'esecuzione di una `read(fd,bytes,&buffer)`

- Il driver del dispositivo inserisce la richiesta nella lista delle richieste pendenti (in attesa che il dispositivo completi l'operazione in corso)
- Al momento di servire la richiesta, il driver alloca un buffer nel kernel per ospitare i dati letti dal dispositivo, trasforma la richiesta in comandi di basso livello per il controllore del dispositivo e programma di conseguenza i registri del controllore
- Il controllore individua l'indirizzo <cilindro,faccia,settore> del blocco del disco, si assicura che il dispositivo di I/O effettui la lettura dei dati e genera un'interruzione quando l'operazione termina
- Tramite il vettore degli interrupt viene attivato il gestore corrispondente che memorizza i dati necessari, invia un ack al driver del dispositivo e termina
- Il driver riceve l'ack, determina quale richiesta di I/O è stata completata, ed il suo stato, e lo segnala al sottosistema di I/O del kernel
- Il kernel trasferisce i dati e/o codici di ritorno nello spazio di memoria dell'invocante, cambia lo stato del processo in ready e sposta il suo PCB dalla coda di attesa sul dispositivo alla coda dei processi pronti
- Quando lo scheduler gli riassegnerà la CPU, il processo invocante potrà riprendere la sua esecuzione

Passi per l'esecuzione di una richiesta di I/O



Gestione delle periferiche di I/O e della memoria secondaria

- Hardware di I/O
- Interazione tra gestore SW e controllore HW
- Compiti ed organizzazione logica del sottosistema di I/O
- Trasformazione delle richieste di I/O in operazioni HW
- Memoria secondaria

Memoria secondaria

- Memoria di massa
- Caratteristiche dei dischi rigidi
- Gestione dei dischi rigidi
- Scheduling dei dischi rigidi

Memoria secondaria

- Memoria di massa
- Caratteristiche dei dischi rigidi
- Gestione dei dischi rigidi
- Scheduling dei dischi rigidi

Memoria di massa

- La memoria di massa è il sistema di memorizzazione non volatile di un computer
- La disponibilità di aree di memoria di massa serve a molte componenti dei SO per, ad esempio, supporto
  - della memoria virtuale dei processi: swap area
  - alla memorizzazione dei file: file system
- La memoria di massa comprende
  - la memoria secondaria, di solito costituita da dischi rigidi e altri dispositivi di memoria non volatile (NVM)
  - la memoria terziaria, non sempre presente, più lenta e più grande, di solito costituita da nastri magnetici, dischi ottici, o spazio di memorizzazione su cloud

Dispositivi di memoria secondaria

- La memoria secondaria può essere classificata in due tipologie distinte:
  - Meccanica: dischi rigidi o magnetici (*hard-disk drive*, *HDD*), dischi ottici, nastri magnetici, …
  - Elettrica (*nonvolatile memory, NVM*): memorie flash, dischi a stato solido (*solid-state disk,* SSD), FRAM, NRAM, …
- La memoria meccanica è generalmente più grande e meno costosa (per byte), ma più lenta della memoria elettrica
- I dispositivi di memoria secondaria più comuni e importanti nei moderni sistemi di elaborazione sono i dischi rigidi e i dispositivi NVM

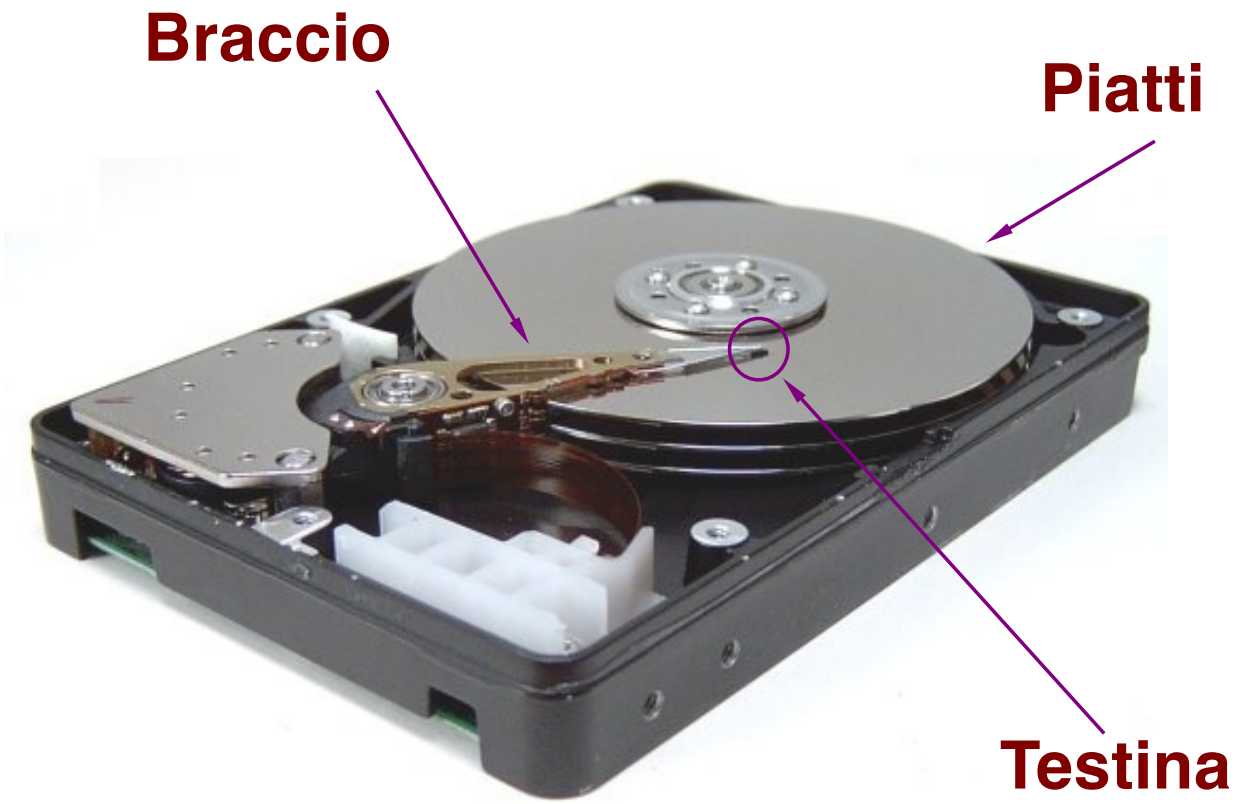
Memoria secondaria

- Memoria di massa
- Caratteristiche dei dischi rigidi
- Gestione dei dischi rigidi
- Scheduling dei dischi rigidi

Struttura fisica dei dischi rigidi

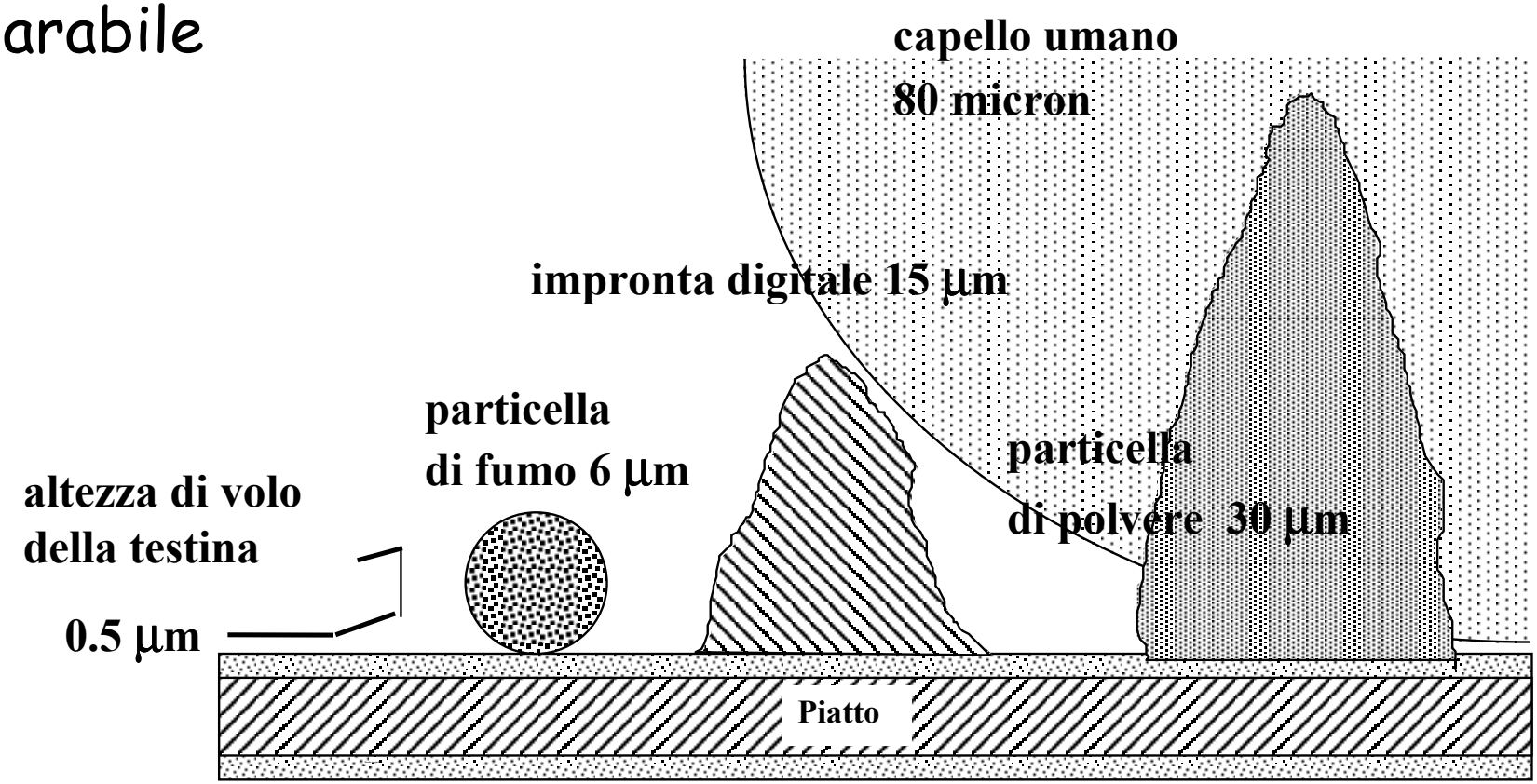
- Un disco rigido è formato da uno o più piatti di plastica, vinile o metallo, rotondi, con diametro compreso tipicamente tra 1,8 e 3,5 pollici
- I piatti sono ricoperti da uno strato di ossido metallico magnetizzabile sul quale i dati possono essere scritti e letti
- Una testina di lettura/scrittura, sorretta da un braccio, modifica lo stato di magnetizzazione del materiale in fase di scrittura e ne rileva lo stato in fase di lettura

Struttura fisica dei dischi rigidi



Struttura fisica dei dischi rigidi

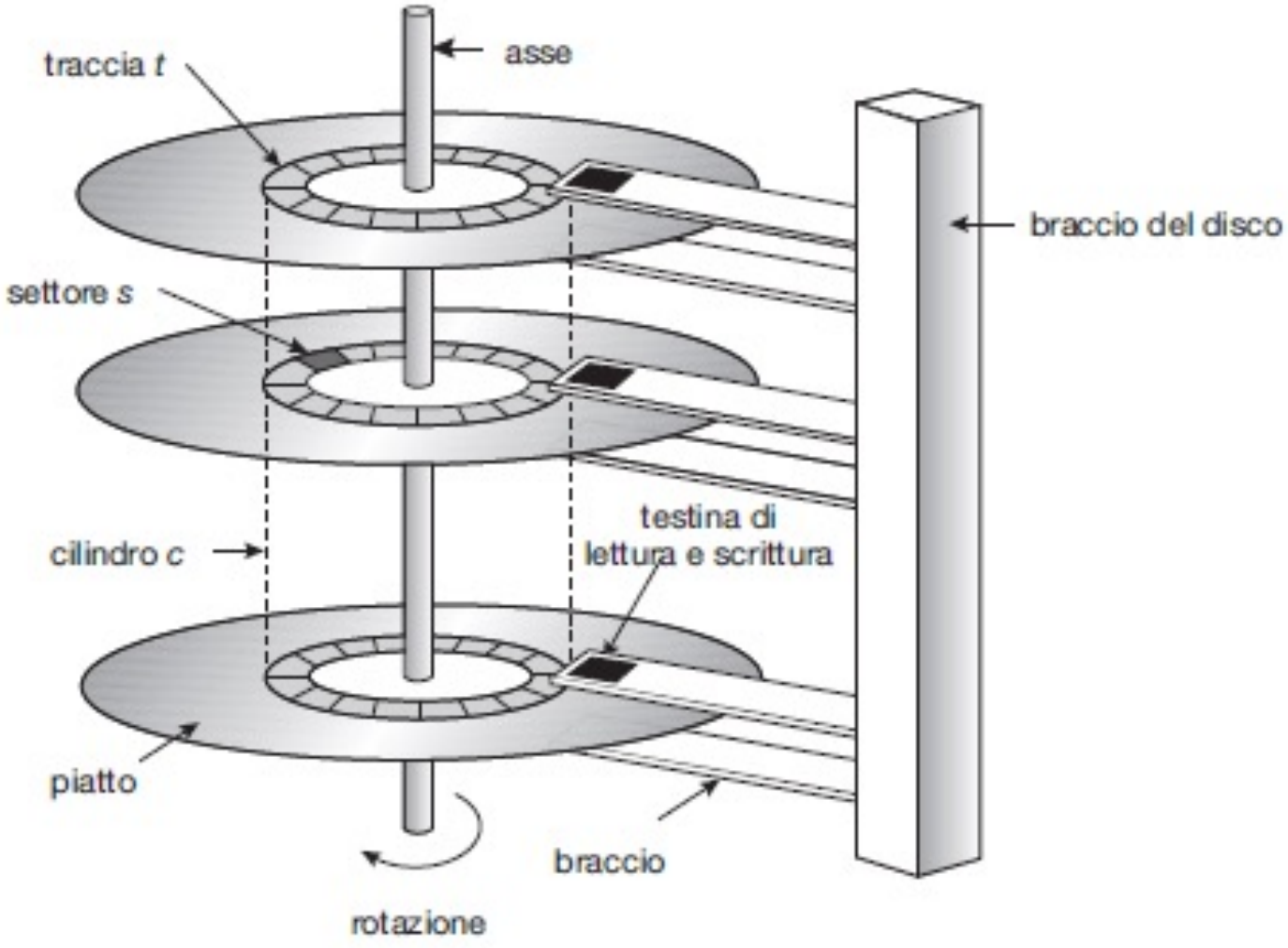
- La testina è sospesa su un cuscino d'aria sottilissimo (altezza di volo, dell'ordine dei micron)
- C'è sempre il pericolo che la testina urti la superficie del disco (crollo della testina) e lo danneggi in maniera irreparabile



Formattazione di basso livello

- Un disco rigido nuovo è tabula rasa: prima che possa memorizzare dati deve essere suddiviso in settori che possano essere letti o scritti dal controllore
  - Tutte le operazioni di I/O sono eseguite alla granularità di settori interi
- La formattazione di basso livello, o formattazione fisica, eseguita solitamente dal produttore, divide il disco in tracce concentriche, a loro volta suddivise in settori, la cui dimensione è solitamente compresa tra 512 byte e 4KB
  - Tra tracce e tra settori adiacenti viene lasciato un minuscolo spazio di separazione, detto gap
- Un insieme di tracce su più superfici equamente distanti dal centro forma un cilindro

Formattazione di basso livello



Settore

È una speciale struttura dati che tipicamente consiste in un preambolo (o intestazione), un'area dati e una coda

- Preambolo e coda contengono dati utilizzati dal controllore del dispositivo
- Il preambolo inizia con un determinato schema di bit che permette all'HW di riconoscere l'inizio del settore
  - Contiene inoltre alcune informazioni, tra cui il numero della traccia e del settore
- La dimensione dell'area dati è determinata dal programma di formattazione a basso livello
- La coda contiene un codice per la correzione degli errori (ECC), cioè informazioni ridondanti che possono essere usate per ripristinare eventuali errori di lettura
  - Spesso è di 16 byte, ma dimensione e contenuto variano da produttore a produttore

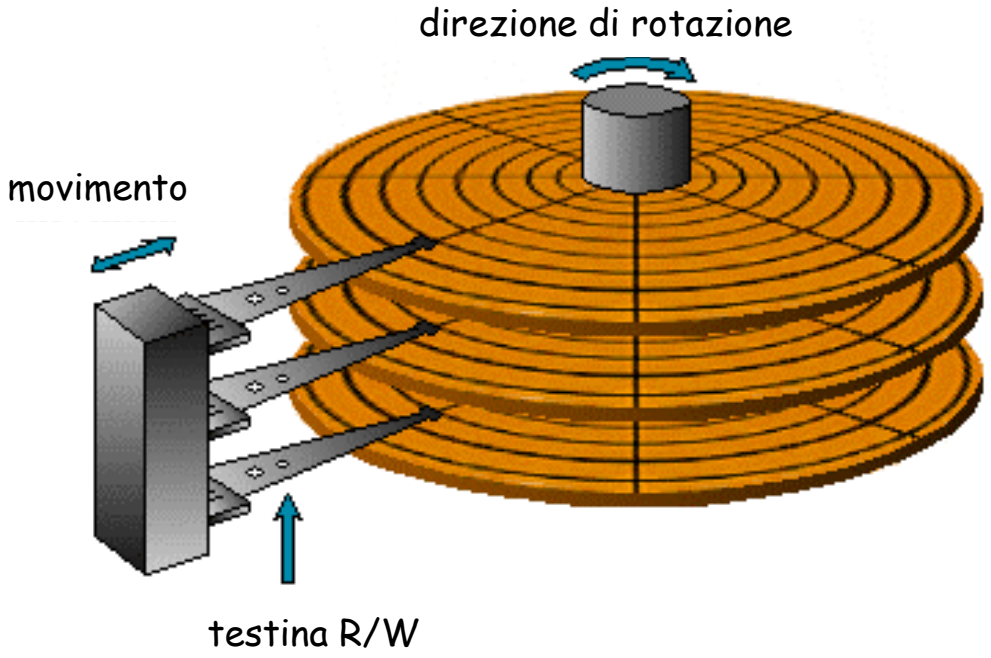
Capacità di un disco

- La tecnologia dei dischi è una di quelle che si evolve più rapidamente
  - La capacità dei dischi si incrementa più velocemente di quanto previsto della legge di Moore per il numero di transistor nei processori (che *raddoppia ogni 18 mesi*)
- In un'unità a disco ci sono migliaia di cilindri concentrici (la densità tipica è di 200.000 tracce ogni pollice di lunghezza del raggio)
- Ogni traccia può contenere centinaia di settori
- Esempio: se un disco rigido si compone di 2 piatti (4 superfici), 16.384 cilindri (16.384 tracce per superficie) e 16 settori, e ciascun settore ha una capacità di 4KB, allora la capacità del disco sarà di 4 × 16.384 × 16 × 4.096 byte, ovvero 4 GB
  - 100GB equivalgono a 50M di pagine testo con doppia spaziatura
- Sono oramai comuni unità a disco con capacità dell'ordine dei TB

Movimenti in un disco

Durante il funzionamento del disco, per posizionarsi sul settore da leggere o scrivere,

- la testina si muove in direzione radiale
- il disco ruota sotto di essa compiendo solitamente tra 60 e 250 giri al secondo
  - Questa velocità viene solitamente espressa in rotazioni per minuto (RPM): velocità tipiche dei comuni dischi sono 5.400, 7.200, 10.000 o 15.000 RPM

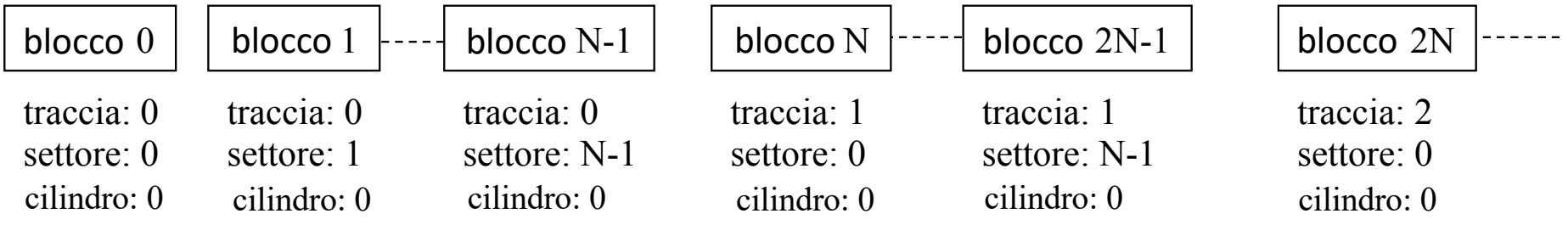


Tipologie di dischi

- Fissi (hard disk) e rimovibili (floppy disk)
- Testine
  - fisse: tante testine di lettura/scrittura quante sono le tracce di una superficie
  - mobili: una sola testina che si muove in direzione radiale al disco
- Velocità di rotazione
  - velocità angolare costante (*constant angular velocity*, CAV)
  - velocità lineare costante (*constant linear velocity*, CLV)
- Connessione al calcolatore
  - Tramite bus di sistema o di I/O, con varie possibili architetture
    - **ATA** (Advanced Technology Attachment), **SATA** (Serial ATA), **eSATA**
    - **SCSI** (Small Computer System Interface), **SAS** (Serial Attached SCSI)
    - **FC** (Fiber Channel)
    - **USB** (Universal Serial bus)
    - **Firewire** (IEEE-1394)
    - …
  - Tramite rete

Geometria virtuale e reale di un disco

- Nei sistemi moderni, al driver (e al SO) tipicamente vengono nascosti i dettagli fisici del disco e viene piuttosto presentata una geometria virtuale: il disco è visto come un grande vettore unidimensionale di blocchi logici
  - Il blocco logico è l'unità di trasferimento dei dati
- La formattazione di basso livello, eseguita solitamente come parte del processo produttivo, stabilisce la geometria reale del disco in termini di cilindri, tracce e settori
- Ogni blocco logico può essere associato ad un settore fisico del disco e l'intero vettore di blocchi logici può essere mappato sui settori del disco



Geometria virtuale e reale di un disco

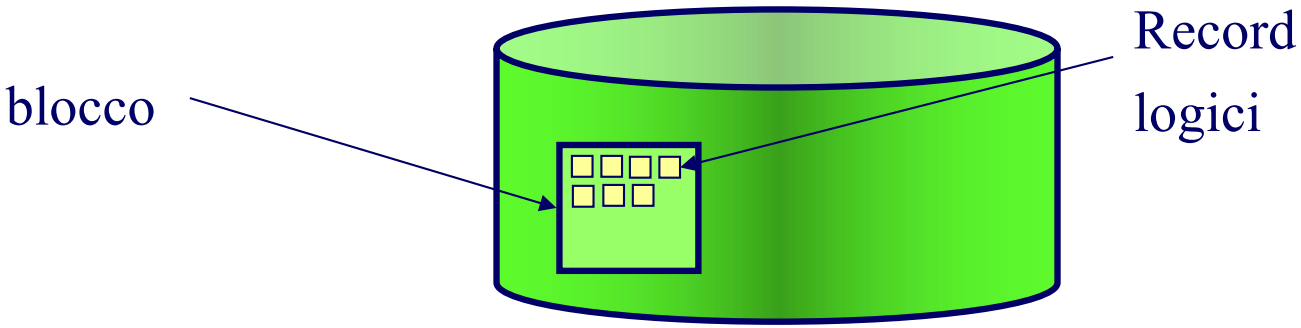
- Un indirizzo di blocco logico è più agevole da utilizzare da parte degli algoritmi rispetto ad un indirizzo fisico del tipo <cilindro,traccia, settore>
  - La mappatura permette di tradurre un indirizzo di blocco logico in un indirizzo di un settore fisico del disco
- In pratica, per il SO la traduzione è difficile da effettuare per tre ragioni
  - Su alcuni dischi, il numero di settori per traccia non è costante
  - Sostanzialmente tutti i dischi hanno settori difettosi che la mappatura nasconde sostituendoli con settori di riserva collocati in altre posizioni sull'unità: l'indirizzo del blocco logico rimane sequenziale, ma la posizione del settore fisico cambia
  - I produttori dei dischi gestiscono la mappatura internamente: stabiliscono la corrispondenza tra blocchi logici e settori correttamente funzionanti al momento della formattazione di basso livello e la fanno gestire al controllore
- Malgrado ciò, gli algoritmi che gestiscono gli HDD tendono a presumere che i blocchi logici siano correlati ai settori fisici cosicché a blocchi vicini corrispondano settori vicini

Blocchi e record logici

- L'unità di trasferimento dei dati nelle operazioni di I/O tra memoria centrale e disco è il blocco (logico) del disco
  - Il blocco logico è anche l'unità di allocazione dei file
- I processi vedono ogni file come una sequenza di record logici
  - Un record logico è l'unità di trasferimento nelle operazioni di accesso al file
- Di solito:

  dimensione blocco >> dimensione record logico

  Quindi ogni blocco contiene diversi record logici contigui



Dimensione dei blocchi: considerazioni

- Una dimensione del blocco grande significa che ogni file, anche se di un solo byte, impegna (almeno) un intero blocco
  - File piccoli sprecano una grande quantità di spazio del disco
- Una dimensione del blocco ridotta significa distribuire la maggior parte dei file su molti blocchi
  - Ciò comporta tempi di ricerca più lunghi e maggiori ritardi di rotazione per leggerli, a scapito delle prestazioni
- Quindi, se l'unità di allocazione è troppo grande si spreca spazio, se è troppo piccola si spreca tempo
  - Per una buona scelta, servono informazioni sulla distribuzione statistica delle dimensioni dei file nel sistema

Memoria secondaria

- Memoria di massa
- Caratteristiche dei dischi rigidi
- Gestione dei dischi rigidi
- Scheduling dei dischi rigidi

Gestione dei dischi: raw disk

- Alcuni SO permettono a certi programmi di vedere un disco, o una sua partizione, a livello di astrazione più basso, e cioè come un vettore monodimensionale di blocchi logici (quindi senza alcuna struttura dati relativa al file system)
  - Tale vettore viene chiamato disco di basso livello (*raw disk*) e l'I/O relativo si chiama I/O di basso livello
- Alcuni sistemi usano un disco di basso livello per l'area di swap dei processi o per la gestione di basi di dati
  - Certi sistemi per la gestione di basi di dati preferiscono questo tipo di I/O perché fornisce più controllo sulla posizione dei record logici nel disco
- L'I/O di basso livello elude tutti i servizi del file system e può essere usato dalle applicazioni per implementare servizi di memorizzazione specializzati più efficienti
  - Ma la maggior parte delle applicazioni preferisce disporre dei servizi forniti da un file system

Gestione dei dischi: file system

- Tipicamente file e directory sono salvati in dispositivi di memorizzazione ad accesso diretto, come dischi fissi, dischi ottici e dischi a stato solido
- I dischi hanno due caratteristiche importanti che ne fanno un mezzo adatto a questo scopo:
  - è possibile accedere direttamente a qualsiasi blocco del disco, risulta quindi semplice accedere a qualsiasi file, sia in modo sequenziale che diretto
  - i blocchi si possono riscrivere localmente: si può leggere un blocco dal disco, modificarlo e quindi riscriverlo nella stessa posizione

Gestione dei dischi: file system

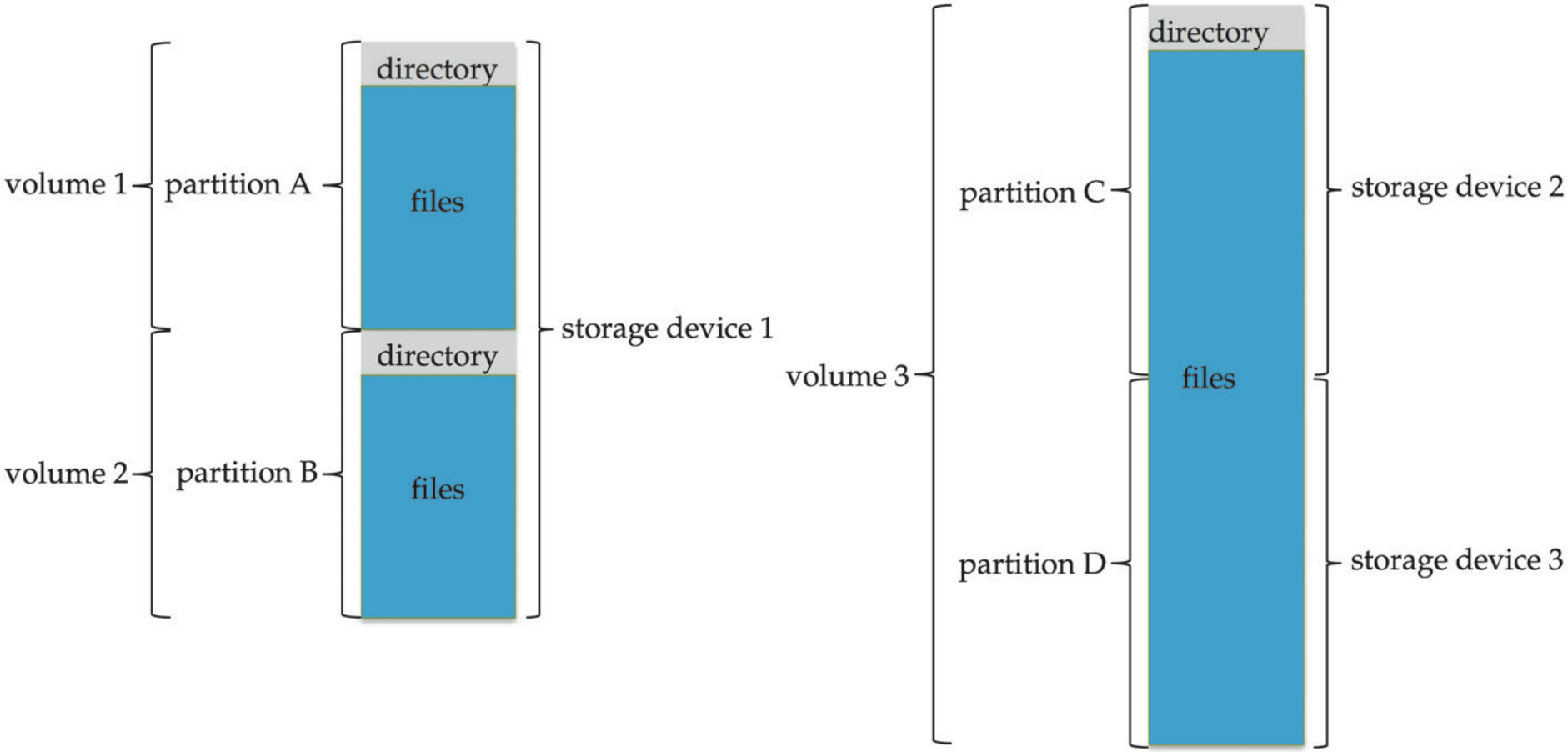
- Per memorizzare i file sul disco, il SO ha prima bisogno di registrare su disco le proprie strutture dati
- Lo fa in 3 passi:
  1. partizionamento: suddivide il disco in uno o più gruppi di cilindri, detti *partizioni*, che poi vengono *montate* per essere rese disponibili al sistema ed agli utenti
     - Una o più partizioni possono essere d'avviamento per il SO; quella da cui si avvia il SO è usata anche per stabilire la root del file system
  2. creazione e gestione del volume: operazione che può essere implicita (es. la partizione coincide con un volume: quel volume è allora pronto per essere montato) o esplicita (es. più partizioni sono usate insieme per creare un unico volume più grande, come in una organizzazione RAID)
  3. formattazione ad alto livello (o logica): creazione vera e propria del file system all'interno di un volume; comporta la scrittura delle strutture dati iniziali del file system, quali struttura della directory, mappe dello spazio libero e allocato, FCB, ...

Partizionamento di dischi

- Un disco potrebbe essere interamente usato per un unico file system, ma per una gestione più agevole è tipicamente suddiviso in uno o più gruppi di cilindri, detti partizioni
- Il partizionamento è utile
  - per limitare la dimensione dei singoli file system
  - per mettere sullo stesso disco file system di tipi diversi
  - per destinare alcune parti del disco ad usi diversi, es. spazio non formattato (*raw partition,* privo di struttura logica) per l'area di swap dei processi
- Le partizioni sono organizzate in volumi
  - Un volume si può pensare come un disco virtuale

Partizioni e volumi

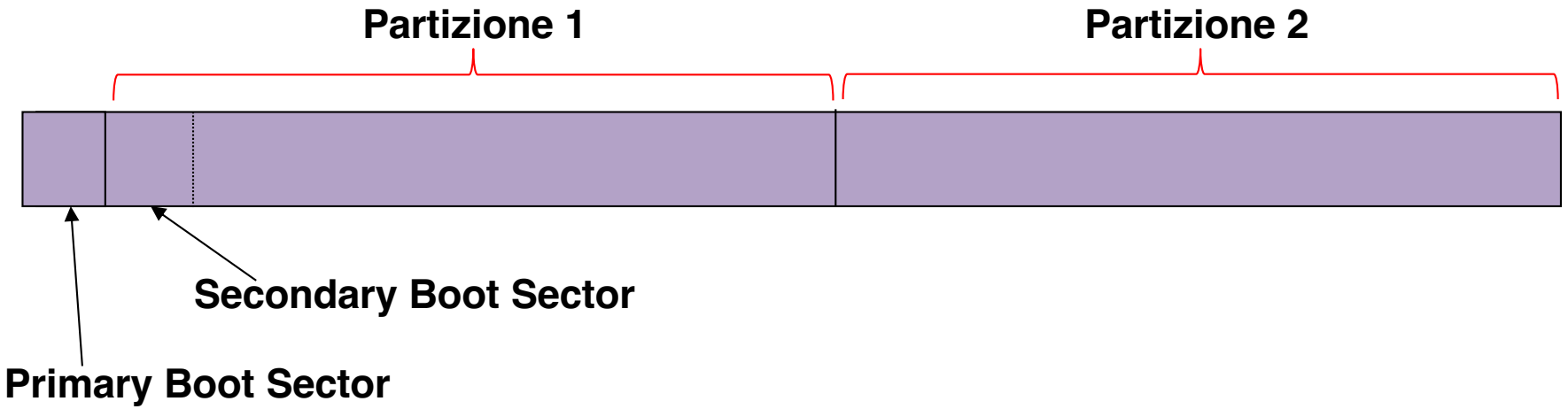
Un volume può coincidere con una partizione o raggruppare più partizioni (eventulamente di dispositivi diversi) in un'unica struttura logica (es. organizzazione RAID)



Volumi

- Ogni partizione/volume può contenere un (solo) file system
  - Il file system organizza e gestisce i blocchi per creare l'astrazione di file e directory insieme con i loro metadati
- Ogni volume contenente un file system contiene anche un indice o directory del volume che registra informazioni, quali nome, posizione, dimensioni e tipo, di tutti i file presenti nel file system contenuto nel volume

Partizionamento di dischi



- Il settore/blocco 0 (o *primary boot sector*) del disco contiene il *Master Boot Record (MBR),* costituito dalla tabella delle partizioni e dal *boot manager* (sequenza di istruzioni necessarie a identificare una partizione attiva e avviare il codice contenuto nel suo *secondary boot sector*)
  - Recentemente MBR è stato rimpiazzato dal *GPT (GUID Partition Table),* uno standard (parte di UEFI) per la definizione della tabella delle partizioni
- Il primo settore/blocco (o *secondary boot sector,* o *boot block*) di una partizione attiva contiene il *boot loader* (codice di avviamento) del SO (è invece vuoto se la partizione non contiene un SO)

Partizioni attive e Montaggio

- Il boot manager può anche essere in grado di interpretare file system diversi e avviare SO diversi
  - I dispositivi possono avere più partizioni attive, ognuna contenente un diverso tipo di file system e un diverso SO
  - Il boot manager può avviare uno dei SO disponibili
- Il boot loader è una sequenza di blocchi contigui che si carica in memoria come un'immagine e la cui esecuzione carica il kernel del SO e ne avvia l'esecuzione
- Durante l'avvio del SO, la partizione utilizzata per l'avvio diviene la partizione radice (root) del file system
  - Contiene il kernel del SO e talvolta altri file di sistema
- A seconda del SO, altre partizioni possono essere 'montate', automaticamente all'avvio o manualmente successivamente, come ulteriori file system

Memoria secondaria

- Memoria di massa
- Caratteristiche dei dischi rigidi
- Gestione dei dischi rigidi
- Scheduling dei dischi rigidi

Prestazioni di un disco rigido

Le prestazioni di un disco sono caratterizzate da due fattori: **TA** & **TT**

- **TA** (tempo di posizionamento o di accesso casuale): tempo necessario per posizionare la testina di lettura/scrittura in corrispondenza del settore desiderato. Nel caso di un disco a testine mobili si ha: **TA = ST + RL**
  - **ST** (tempo di ricerca o *seek time*): tempo necessario a spostare il braccio del disco in corrispondenza della traccia/cilindro desiderato

    ST dipende dalla velocità con cui si muove il braccio e dalla distanza tra la traccia su cui è posizionata la testina e quella su cui dovrà spostarsi (2-10ms)
  - **RL** (latenza di rotazione o *rotational latency*): tempo necessario affinché il settore desiderato si porti, tramite la rotazione del disco, sotto la testina

    RL, mediamente, è il tempo impiegato per compiere mezzo giro (4ms a 7.500 RPM, 3ms a 10.000 RPM, …)
- **TT** (tempo di trasferimento): tempo necessario per effettuare il vero e proprio trasferimento dei dati relativi ad un singolo settore
  - Si può approssimare con il quoziente della divisione tra il tempo per compiere un giro (tempo di rotazione) ed il numero di settori per traccia

  Dipende dalla velocità con cui dati passano sotto la testina di lettura/scrittura, quindi dalla velocità di rotazione e dalla densità del disco, ma anche dall'interfaccia tra disco e calcolatore

  È solitamente dell'ordine dei microsecondi

Prestazioni di un disco rigido

- La formattazione a basso livello influisce anche sulle prestazioni (in particolare sul tempo di trasferimento) perché determina la densità del disco
  - Un disco da 10.000 RPM impiega 6ms ad effettuare una rotazione
  - Se ci sono 300 blocchi per traccia di 512 byte ciascuno, ci sono allora 153.600 byte in una traccia, che verranno letti in 6ms
  - Quindi procede a una velocità di 24,4MB/s, a prescindere dall'interfaccia (anche con SCSI a 640MB/s)
- Comunque, essendo il tempo di trasferimento dell'ordine dei microsecondi, le prestazioni dipendono principalmente dal tempo di ricerca ST e dalla latenza di rotazione RL
- Le prestazioni influenzano
  - il tempo che un processo resta in attesa nella coda del dispositivo (cioè il tempo che intercorre fra la richiesta di servizio ed il completamento dell'ultimo trasferimento)
  - l'ampiezza di banda (numero di byte trasferiti nell'unità di tempo)

Migliorare le prestazioni

- Criteri con cui memorizzare i dati su disco
  - Es. allocazione dei blocchi di un file: contigua vs. concatenata/indicizzata
    - Un programma che accede sequenzialmente un file, se questo è allocato in modalità contigua, genererà molte richieste raggruppate, con un conseguente limitato spostamento del braccio
    - Se il file è invece allocato in modalità concatenata o indicizzata, allora potrebbe includere blocchi sparsi per tutto il disco e richiedere quindi un maggiore spostamento del braccio
  - Oltre al metodo di allocazione dei file anche la posizione di tabelle di directory e FCB è importante
- Criteri con cui schedulare le richieste di accesso al disco

Schedulazione degli accessi al disco

- Le richieste di operazioni di I/O sui dispositivi di memoria secondaria sono generate sia dal file system che dal sistema di memoria virtuale
- Una richiesta contiene diverse informazioni, quali
  - tipo di operazione: input o output
  - indirizzo del disco per il trasferimento (sotto forma di uno o più blocchi logici)
  - indirizzo di memoria per il trasferimento
  - numero di blocchi da trasferire
- Se l'unità a disco desiderata ed il relativo controllore sono disponibili, la richiesta si può soddisfare immediatamente
- Altrimenti, la richiesta viene aggiunta alla coda delle richieste pendenti per quell'unità
- Il driver del dispositivo ha così modo di migliorare le prestazioni ordinando opportunamente le richieste in coda

Schedulazione degli accessi al disco

- Sono stati proposti vari algoritmi per determinare in che ordine servire le richieste di I/O verso un disco rigido che sono in attesa su una coda
  - FCFS: First Come First Served
  - SSTF: Shortest SeekTime First
  - SCAN o algoritmo dell'ascensore
  - C-SCAN: Circular SCAN
  - LOOK e C-LOOK
- Per confrontarli useremo la distanza totale (in termini di numero di cilindri visitati) coperta dalla testina per servire tutte le richieste in coda
- Supponiamo di avere un disco con 200 cilindri/tracce (0-199) e la seguente coda di richieste di accesso ai cilindri

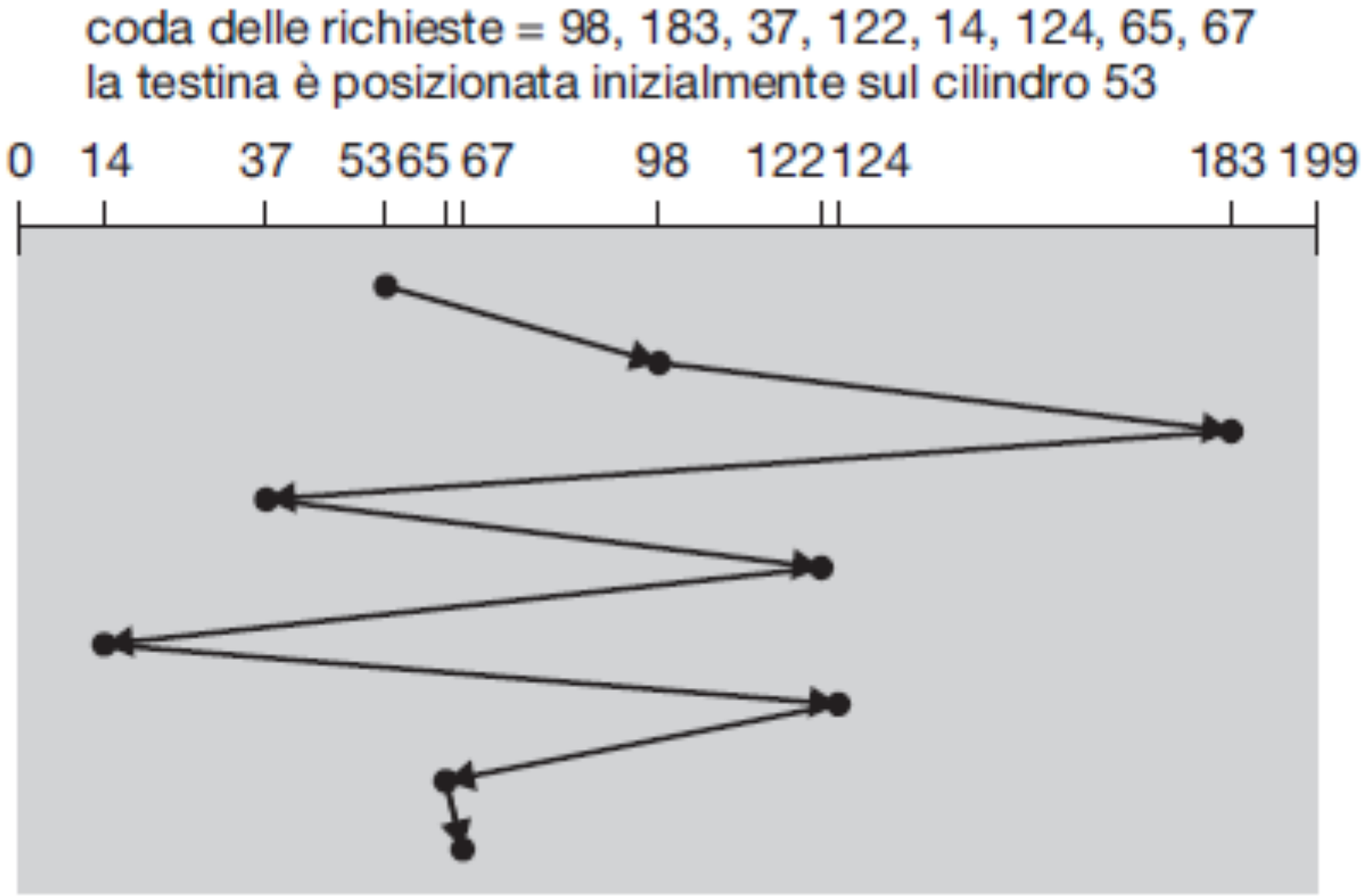
  98, 183, 37, 122, 14, 124, 65, 67

  Supponiamo anche che inizialmente la testina di lettura/scrittura sia posizionata sul cilindro 53

FCFS: First Come, First Served

- Le richieste sono servite nell'ordine con cui sono accodate
- È intrinsecamente equo, quindi evita starvation
- Non corrisponde ad alcun criterio di ottimalità, non garantisce la massima velocità del servizio

FCFS: First Come, First Served

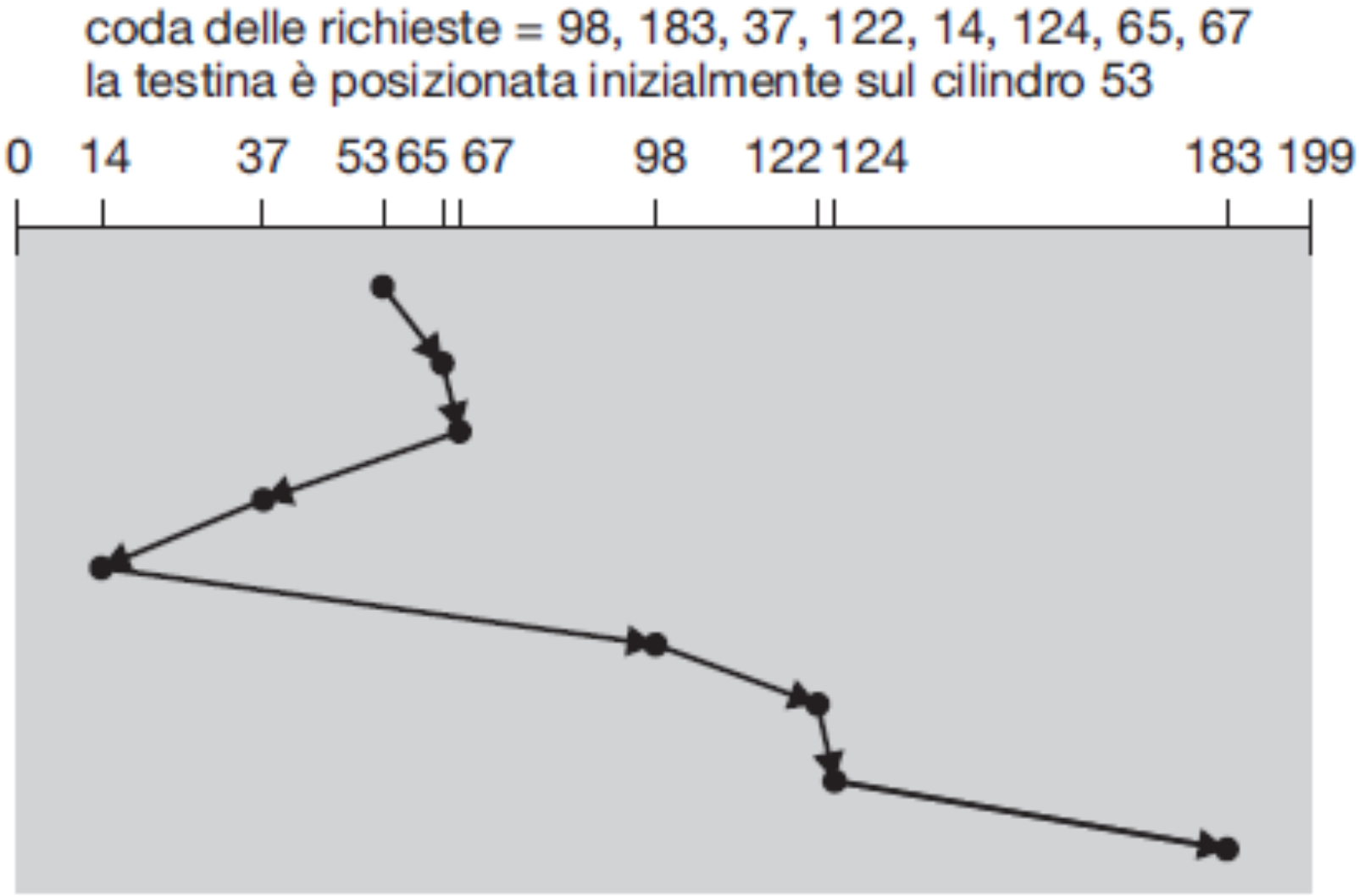


Distanza totale = 640

SSTF: Shortest SeekTime First

- Seleziona la richiesta che fa riferimento alla traccia più vicina a quella corrente
  (è essenzialmente una forma di schedulazione SJF)
- Minimizza il tempo di ricerca (cioè il tempo di posizionamento della testina sulla traccia)
- Può causare starvation di alcune richieste

SSTF: Shortest SeekTime First



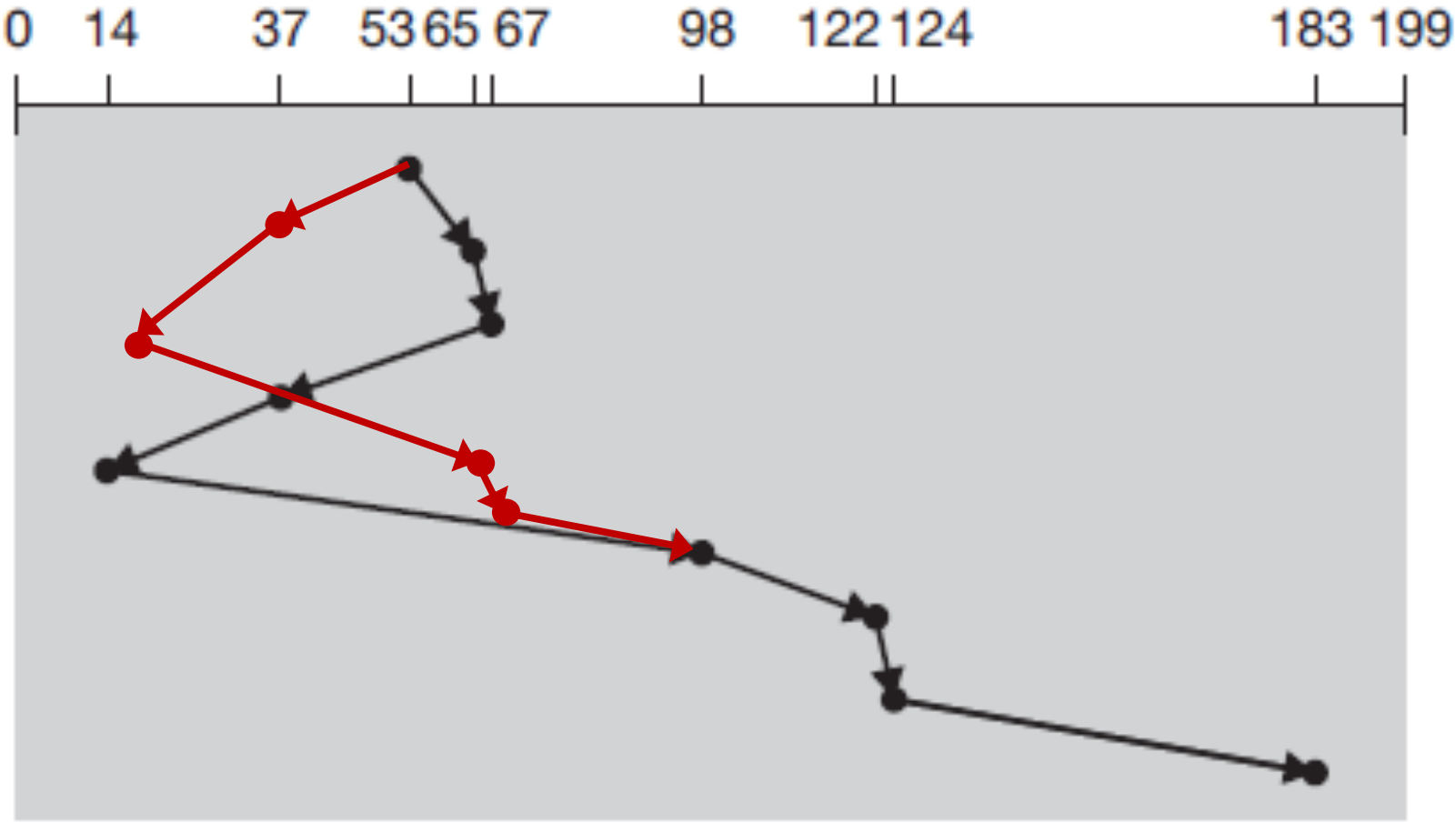
Distanza totale = 236

SSTF: osservazioni

- Non è detto che scegliere la prossima richiesta pendente da servire in modo che il tempo di ricerca sia minimo risulti poi in una minimizzazione del tempo complessivo di ricerca

SSTF: Shortest SeekTime First

Es. servendo prima 37, anzicché 65, e poi 14, 65, 67, 98, 122, 124 e 183, la distanza totale diventa 208, anzicché 236

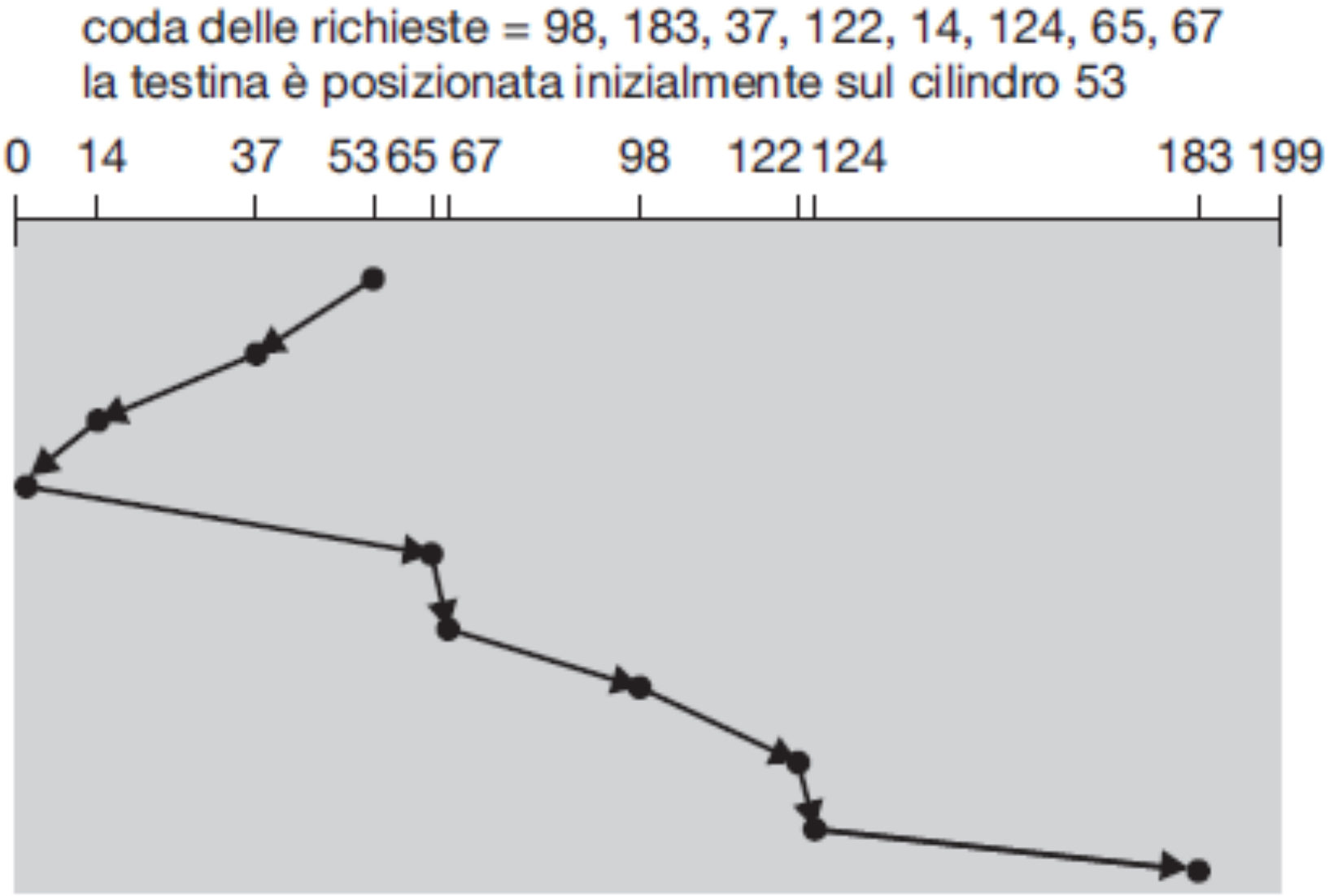


Distanza totale = 208

SCAN (o algoritmo dell'ascensore)

- Il braccio del disco si muove da una estremità all'altra del disco servendo le richieste man mano che raggiunge ogni traccia, finchè non arriva all'estremità del disco, nel qual caso inverte la marcia

SCAN (o algoritmo dell'ascensore)



Distanza totale = 236

SCAN (o algoritmo dell'ascensore)

- Inconveniente: se arriva una richiesta di accesso ad una traccia adiacente alla posizione corrente ma dalla parte opposta al movimento della testiva, tale richiesta deve attendere finchè la testina arriva all'estremità, inverte la marcia, e serve tutte le richieste che la precedono

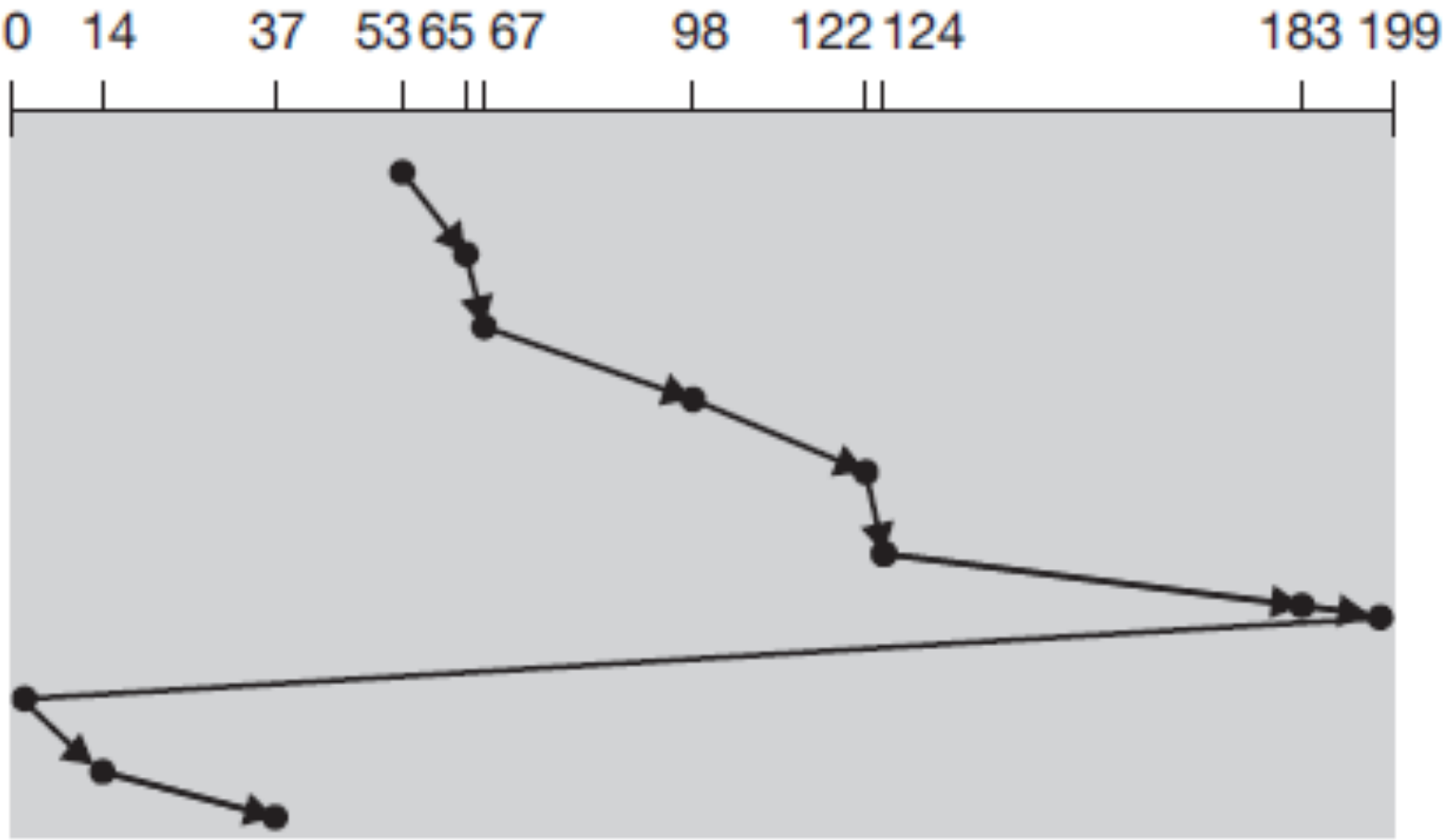
C-SCAN o SCAN circolare

- Variante di SCAN progettata per fornire un tempo di attesa più uniforme
- La testina si muove da una estremità all'altra del disco, servendo le richieste lungo il percorso

  Quando la testina raggiunge una estremità ritorna direttamente all'altra estremità del disco, senza servire alcuna richiesta durante il ritorno
- In pratica, tratta i cilindri come una lista circolare che si riavvolge dal cilindro finale al primo

C-SCAN o SCAN circolare

coda delle richieste = 98, 183, 37, 122, 14, 124, 65, 67
la testina è posizionata inizialmente sul cilindro 53



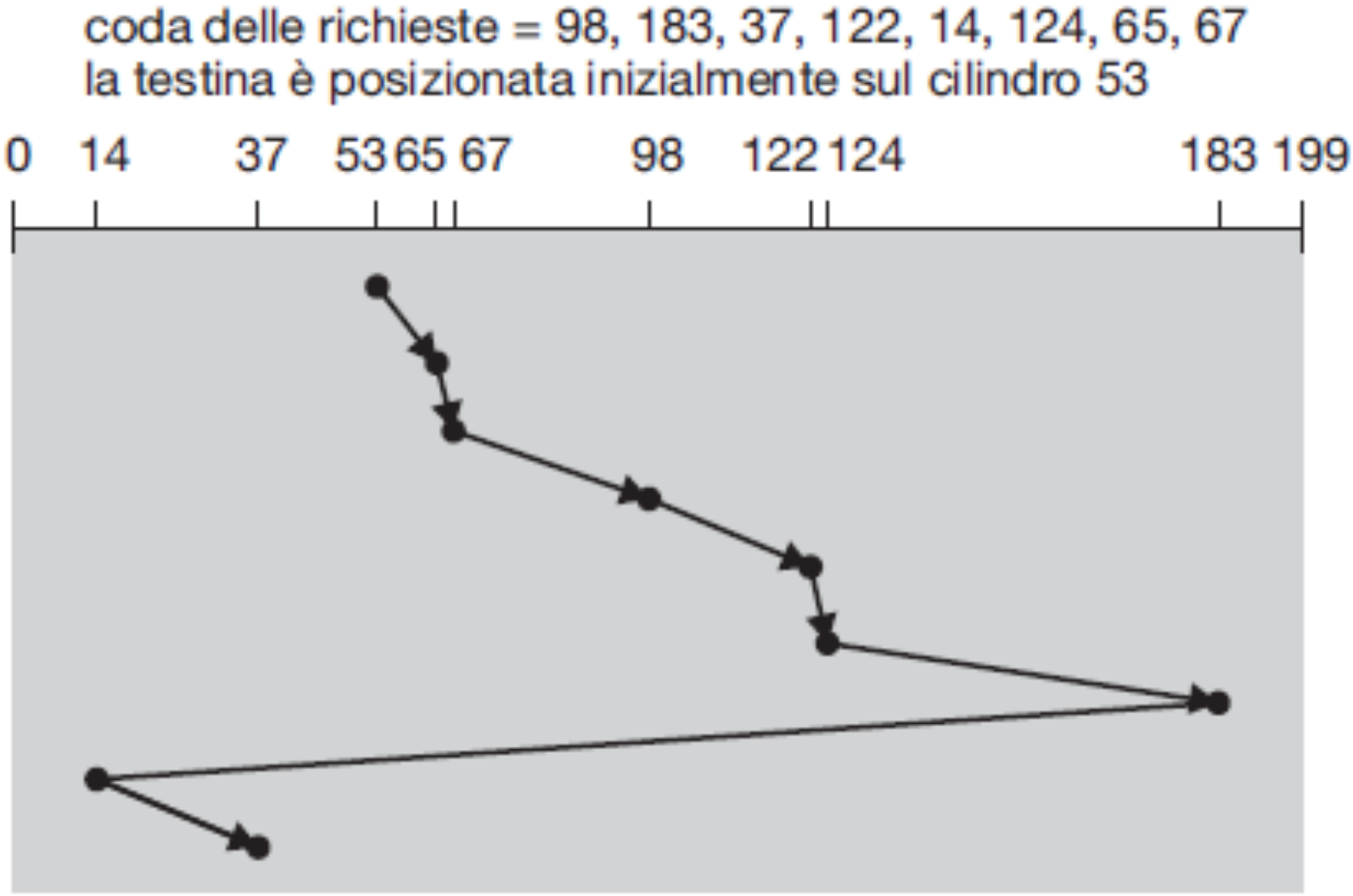
Distanza totale = 382

Schedulazione LOOK e C-LOOK

- Varianti di SCAN e C-SCAN, rispettivamente
- Il braccio arriva fino alla richiesta finale rispetto alla direzione di movimento, e non fino all'estremità del disco

  Lì inverte immediatamente la propria direzione, senza percorrere tutta la strada sino all'estremità del disco

Schedulazione C-LOOK



Distanza totale = 322

Considerazioni sugli algoritmi di schedulazione

- Gli algoritmi di schedulazione delle richieste verso un disco rigido danno tacitamente per scontato che la geometria reale del disco sia la stessa di quella virtuale
- Nel caso ciò non si verifichi, i criteri su cui si basano non hanno senso, perché il SO non può essere certo della reale distribuzione dei settori interessati e, quindi, delle loro relazioni di vicinanza
- Tuttavia, se il controllore del disco può accettare più richieste in attesa, può usare internamente gli algoritmi di schedulazione, che restano quindi validi a un livello più in basso (quello del controllore)

Considerazioni sulle prestazioni

- Con i dischi rigidi moderni il tempo di ricerca e la latenza di rotazione sono così predominanti sulle prestazioni che la lettura di uno o due settori per volta è molto inefficiente
- Perciò, molti controllori leggono e gestiscono la memorizzazione temporanea di più settori, anche quando ne è richiesto uno solo
  - Qualunque richiesta di lettura, causa la lettura di quel settore e di buona parte della traccia su cui si trova, a seconda di quanta cache è disponibile nella memoria del controllore