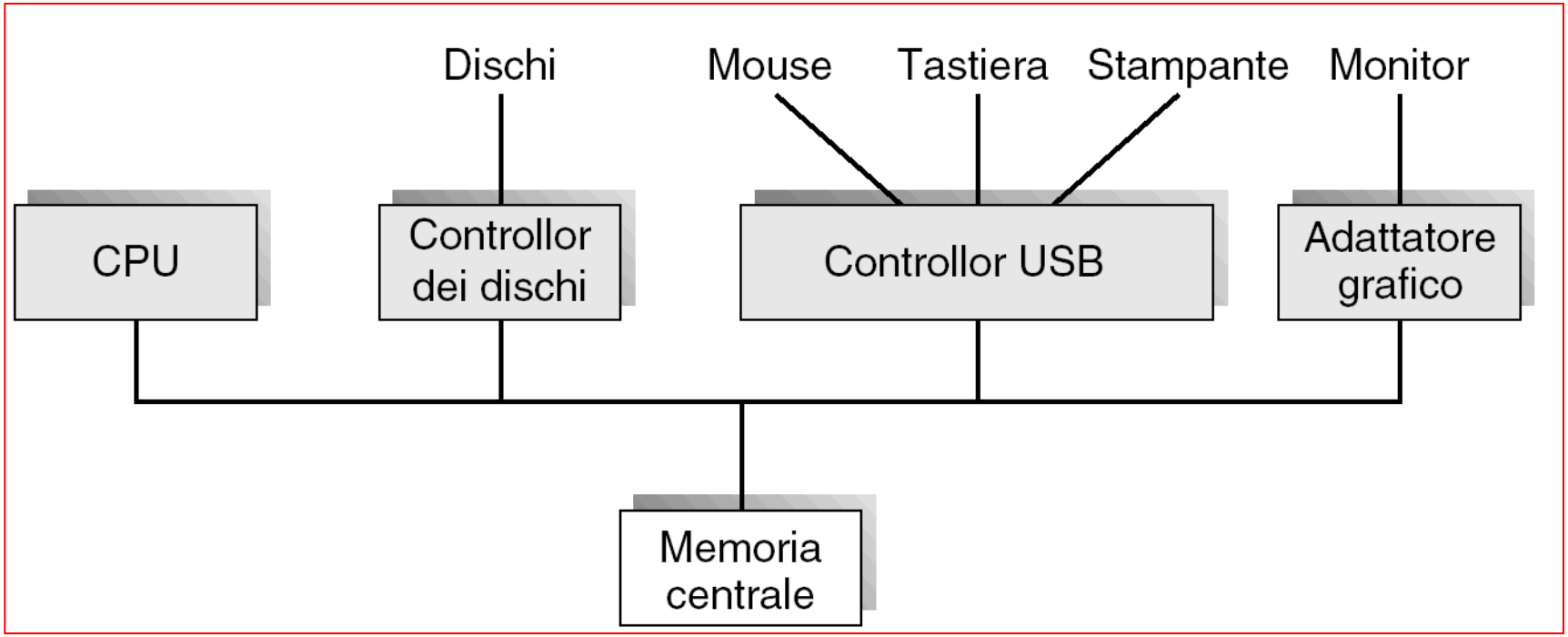
Capitolo2: Architettura del calcolatore

- Funzionamento di un calcolatore general purpose.
- Struttura dell'input/output.
- Struttura della memoria.
- Gerarchia di memorizzazione.
- Protezione hardware.

Struttura generale di un PC



Funzionamento del calcolatore

- La CPU e i controller dei dispositivi di I/O possono operare in modo concorrente.
- Ogni controller è incaricato di gestire una specifica periferica o un set di periferiche similari.
- Ogni controller ha una memoria temporanea locale (*buffer*).
- La CPU sposta i dati da/per la memoria principale a/per i buffer locali.
- L'Input procede dal dispositivo al buffer locale del controller.
- L'Output procede dal buffer locale del controller al dispositivo.
- Il controller del dispositivo informa la CPU che ha concluso la sua operazione generando una interruzione (*interrupt).*

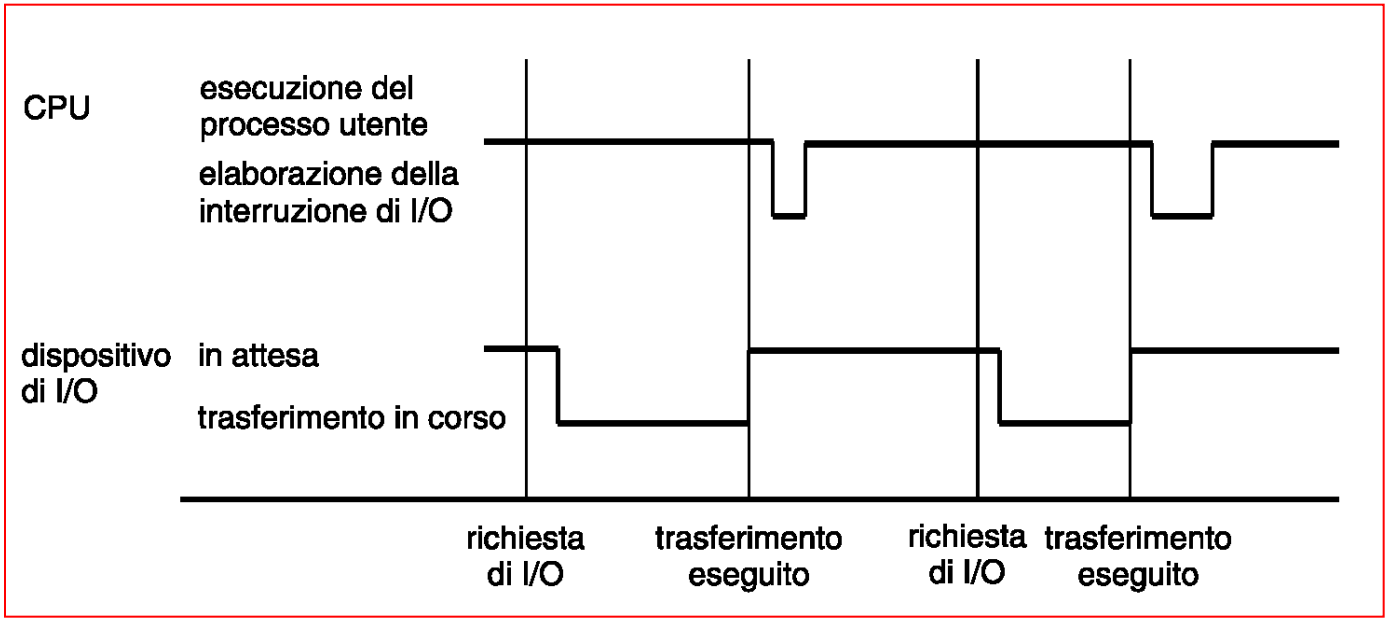
Comuni funzioni degli Interrupt

- L'interrupt trasferisce il controllo della CPU alla routine di gestione interrupt, generalmente attraverso un "*vettore di interrupt*", che contiene gli indirizzi di tutte le routine di servizio.
- Il meccanismo di gestione interrupt deve salvare l'indirizzo dell'istruzione interrotta e lo stato del processore.
- Gli altri interrupt (relativi alle periferiche) sono disabilitati (*mascheramento)* mentre un interrupt viene trattato dal sistema, per prevenire la perdita di segnali.
- Una *trap* è un interrupt generato dal software e può essere causato da un errore (p. es. una divisione per zero) o da una richiesta utente (operazioni di I/O).
- I S.O. moderni sono guidati dalle interruzioni (*interrupt driven*).

Gestione degli interrupt

- Il sistema operativo preserva lo stato della CPU memorizzando i registri ed il *program counter*.
- Determina il tipo di interrupt ricevuto:
  - *polling.*
  - sistema a *interrupt vettorizzato.*
- Porzioni di codice separate determinano quali azioni devono essere eseguite per ogni tipo di interrupt.

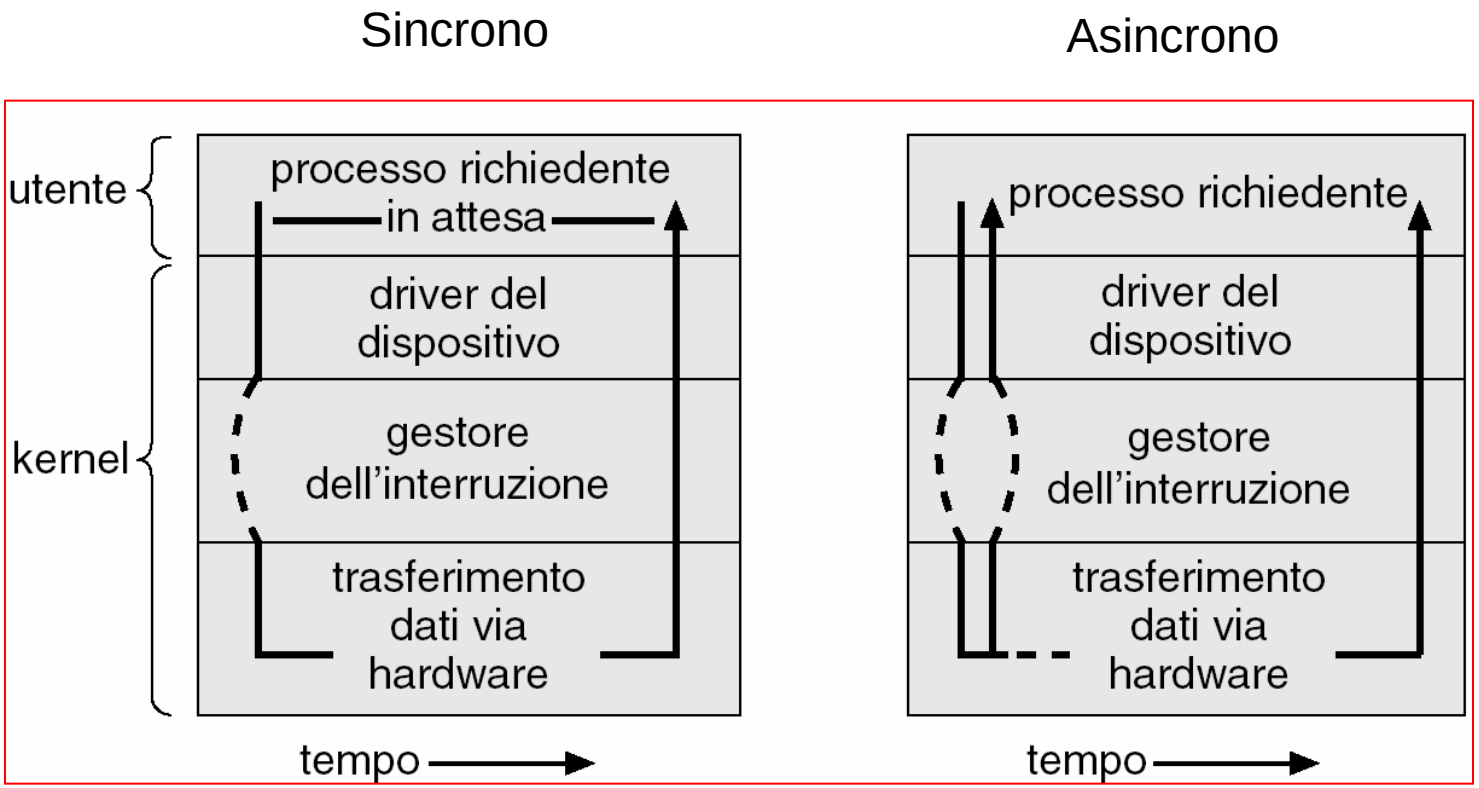
Esecuzione di un interrupt per un singolo processo in uscita



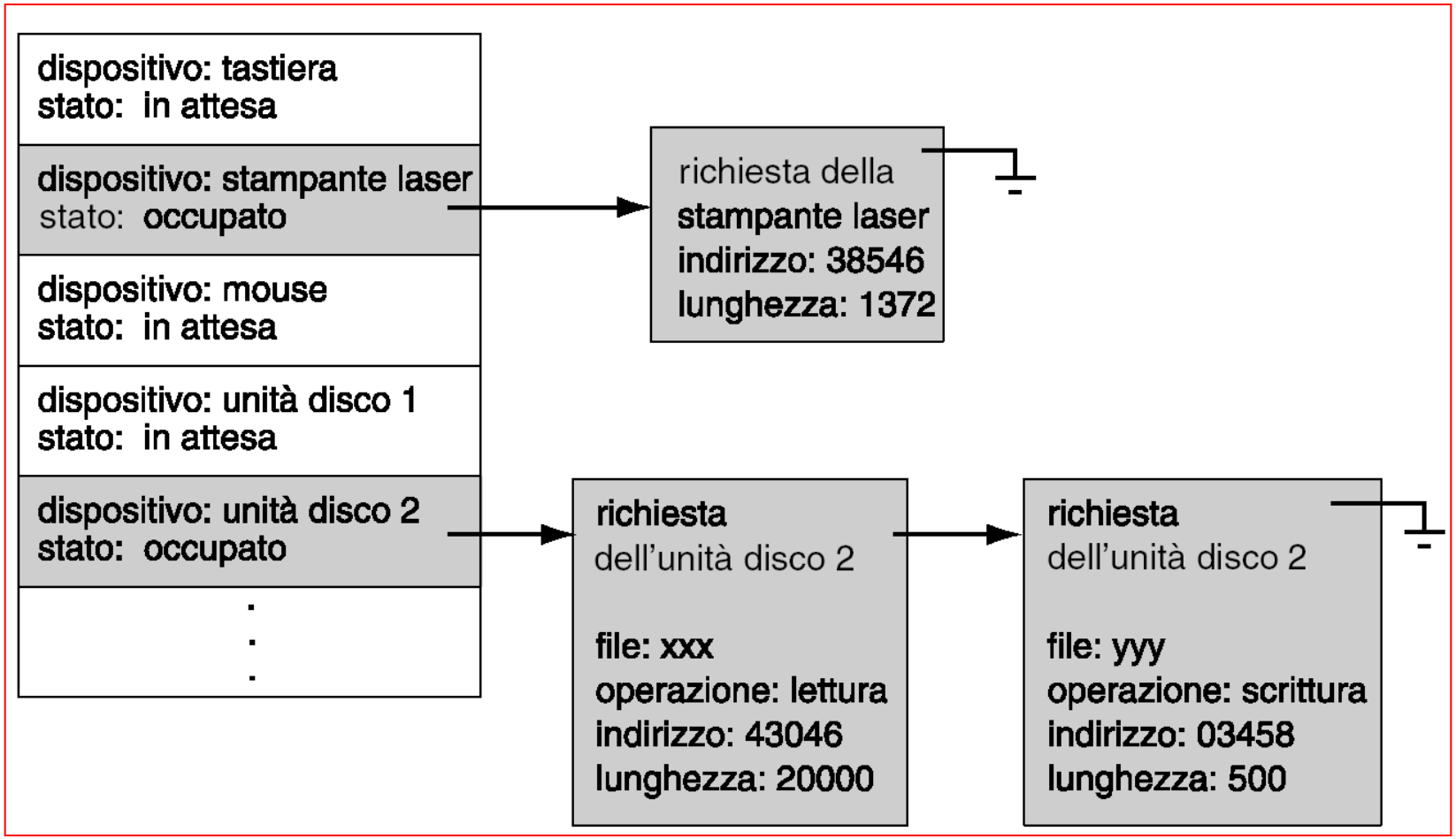
Struttura dell'input/output

- **I/O Sincrono** – la gestione viene iniziata e al suo completamento il controllo viene restituito al processo utente.
  - L'istruzione di *wait* rende la CPU inattiva fino all'interrupt successivo (interrupt di ritorno).
  - Ciclo di attesa (*Wait loop*): `LOOP: jmp LOOP`
  - Solo una richiesta di I/O alla volta può essere tenuta in sospeso, non processi di I/O simultanei.
- **I/O Asincrono** – la gestione viene iniziata e il controllo viene restituito al processo utente senza attendere il completamento dell'operazione.
  - Chiamata di sistema (*System call)* – richiesta al S.O. per far sì che un processo utente possa attendere il completamento dell'I/O.
  - *Device-status table* contiene un valore di ingresso per ogni dispositivo di I/O che ne indica il tipo, l'indirizzo e lo stato.
  - Il sistema operativo esegue una ricerca nella tabella del dispositivo di I/O per determinarne lo stato e modifica il valore in ingresso per rispecchiare l'interrupt.

I due metodi di I/O



Device status table



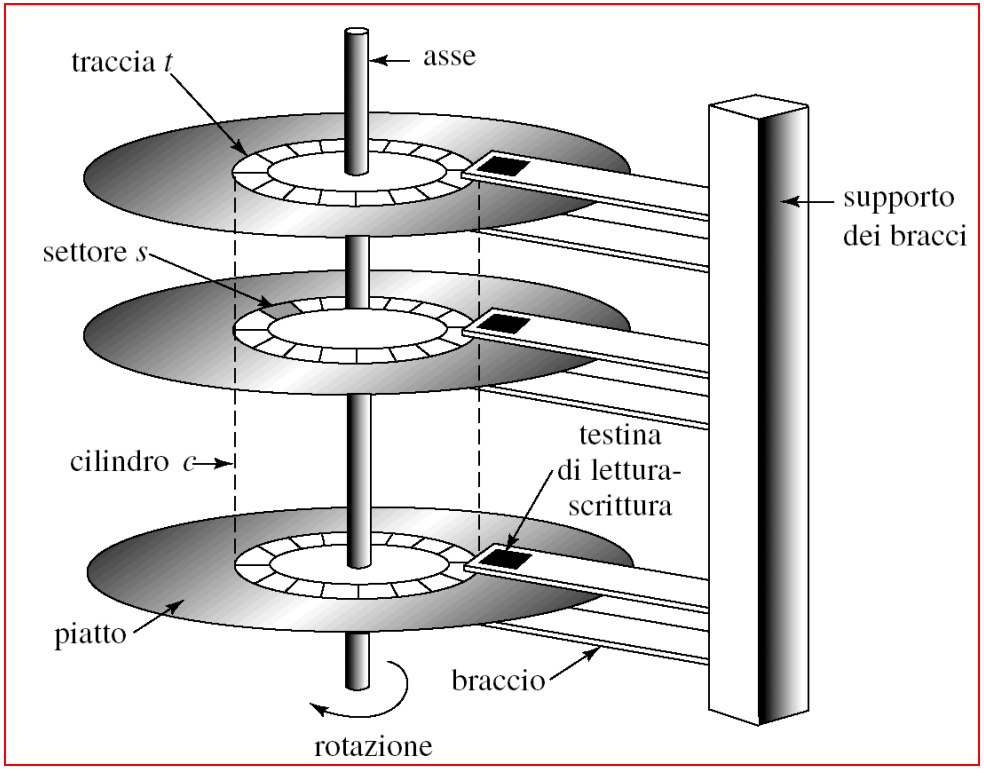
Struttura del DMA

- DMA = Direct Memory Access.
  - Una tastiera trasferisce 1 byte ogni 1000 μsec, una tipica routine di interruzione per I/O occupa 2 μsec di tempo di CPU. Quindi la CPU è in idle per i restanti 998 μsec.
  - Un HD o un network ad alta velocità trasferiscono 1 byte ogni 4 μsec. La CPU è dunque in idle per 2 μsec. Quindi non resta molto tempo per l'esecuzione del processo.
- Usato per i dispositivi in I/O ad alta velocità in grado di trasmettere informazioni a velocità prossime a quella della memoria.
- Funzionamento di base:
  - Un programma o il S.O. richiedono un trasferimento di dati.
  - Il S.O. identifica il buffer per il trasferimento.
  - Il modulo DEVICE DRIVER del S.O. inizializza i registri del controller del DMA
    - Identificazione della periferica sorgente/destinazione.
    - Identificazione degli indirizzi di memoria destinazione/sorgente.
- Il controller del dispositivo trasferisce un intero blocco di dati direttamente al/dal proprio buffer da/in memoria, senza alcun intervento da parte della CPU.
- Viene generato soltanto un interrupt per blocco, invece che un interrupt per ogni byte.

Struttura della memoria

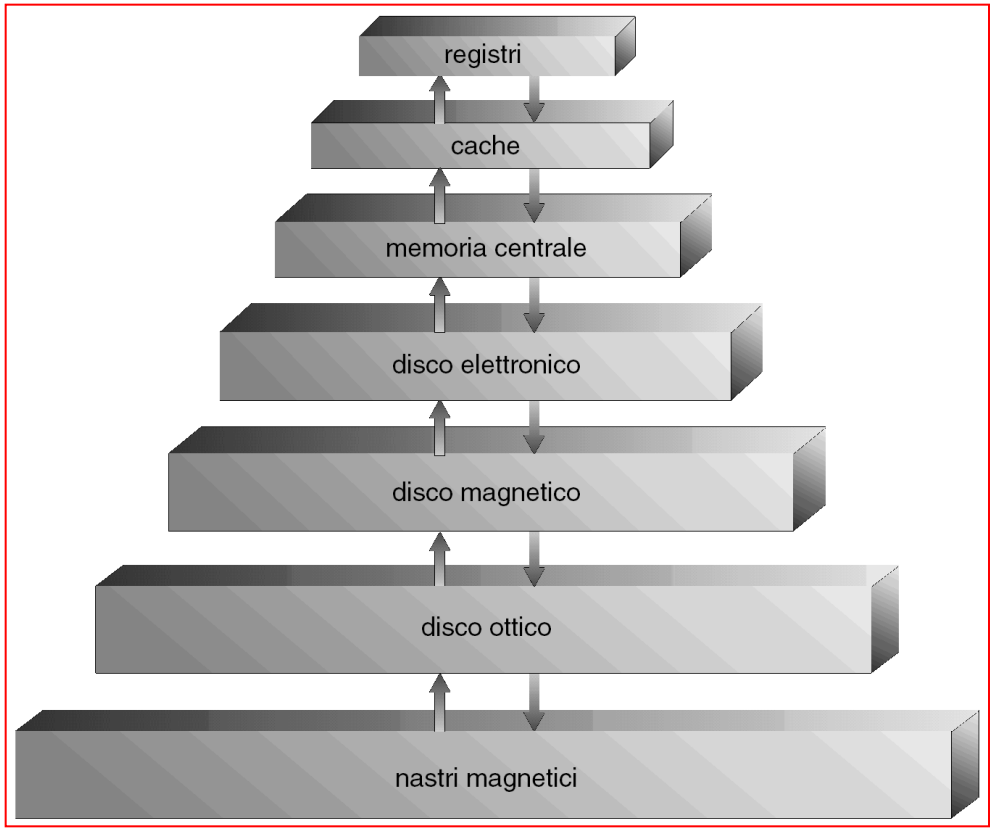
- *Memoria centrale* – unica unità di memorizzazione cui la CPU può accedere direttamente via bus.
  - Indirizzamento alla word.
  - Interazione mediante sequenze di primitive `load/store` che spostano dati da/verso la memoria centrale a/dai registri della CPU.
    - **I/O mappato in memoria** (blocchi di indirizzi di memoria riservati sono mappati in registri dei controller dei dispositivi di I/O).
    - **I/O programmato** (la CPU interroga un bit di controllo relativo ad una data PORTA di I/O per scegliere l'istante di trasferimento dei dati)
    - **I/O interrupt driven** (la CPU riceve un interrupt quando il device è pronto)
- *Memoria secondaria* – estensione della memoria centrale che è in grado di mantenere grandi quantità di dati in modo permanente.
- *Dischi magnetici* – piatti con profilo piano circolare in metallo rigido o vetro ricoperti da materiale ferromagnetico. Diametri da 1.8" a 5.25".
  - La superficie di un piatto è divisa in modo logico in *tracce*, che sono suddivise in *settori.*
  - Il controller dei dischi (*disk controller)* determina l'interazione logica tra il dispositivo e il computer.

Struttura di un disco magnetico



- Velocità rotazionali da 5400 a 7200 rpm. Il tempo di accesso dipende da:
  - Transfer rate.
  - Tempo di posizionamento:
    - Search time (tempo per raggiungere il ciclindro desiderato).
    - Latenza rotazionale (tempo per raggiungere il settore desiderato).
- Un drawback molto comune è l'*head crash*.
- EIDE, ATA, SCSI sono tecnologie di bus standard per l'interfacciamento verso dischi magnetici.
- Tipicamente i disk controller usano meccanismi memory mapped dell'I/O

Gerarchia di memorizzazione

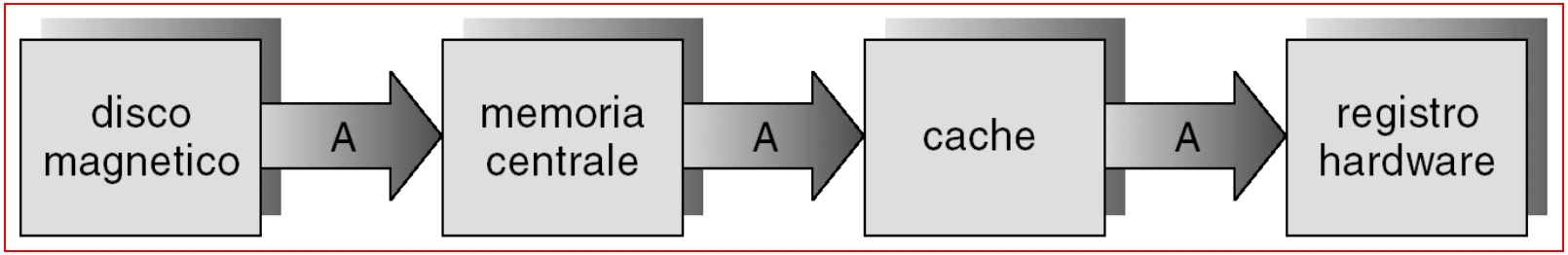


- I sistemi di memorizzazione sono organizzati a livello gerarchico in base a:
  - Velocità/tempo di accesso.
  - Costo/dimensioni.
  - Volatilità.
- *Caching* – copia temporanea dell'informazione in un sistema di memorizzazione più veloce.
- La memoria centrale può essere vista come *cache* veloce per dispositivi di memorizzazione secondari.

Caching

- Uso di una memoria ad alta velocità per mantenere i dati ad accesso recente.
- Necessita di una politica di *gestione della cache.*
- Il movimento delle informazioni tra i livelli della gerarchia di memorizzazione può essere sia esplicito che implicito.
- Il caching introduce un altro livello nella gerarchia dei dispositivi di memorizzazione.
  - Questo richiede che i dati memorizzati simultaneamente in più di un livello siano *coerenti* e *consistenti.*

Migrazione di un dato A dall'HD ai registri HW

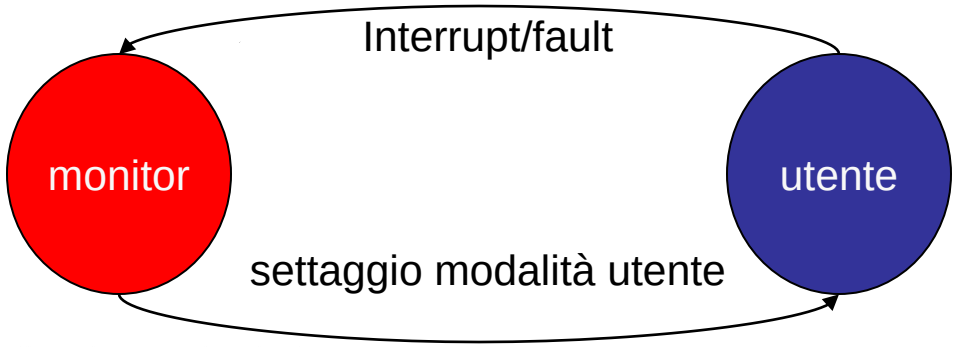


Protezione Hardware

- La multiutenza, la multiprogrammazione e la condivisione di risorse implicano:
  - Un più efficiente sfruttamento del sistema.
  - L'insorgenza di problemi derivanti dalla propagazione di bachi tra processi diversi.
  - Sistemi operativi con funzioni di controllo residente.
- Nei sistemi multiprogrammabili determinati errori potrebbero influenzare programmi in maniera incrociata ed addirittura inficiare il funzionamento del monitor residente.
  - Funzionamento in modalità differenziate.
  - Protezione dell'I/O.
  - Protezione della memoria.
  - Protezione della CPU.
- In assenza di protezione HW, un sistema dovrebbe eseguire solo un processo per volta altrimenti tutti i dati in uscita potrebbero essere considerati "sospetti".
- Un S.O. ben progettato deve assicurarsi che un programma errato o malizioso non possa indurre errori incrociati.

Modalità differenziate

- Condividere le risorse comporta la necessità di proteggere il sistema operativo da qualsiasi programma funzionante in modo scorretto.
- Molti sistemi operativi forniscono un supporto hardware che permette di distinguere vari modi di esecuzione:
  1. Modalità utente (*User mode)* – attività eseguita dall'utente.
  2. Modalità di controllo *(Monitor mode, ma anche kernel mode* o *system mode*) – attività eseguita dal sistema operativo.
- Un bit di modalità *(mode bit)* aggiunto all'hardware del computer per indicare la modalità corrente: monitor (0) o user (1).
- Quando si presenta una trap o un interrupt l'hardware commuta dalla modalità utente a quella di controllo.



Le *istruzioni privilegiate possono essere* eseguite solo in modalità di controllo (*System call*).
All'avvio, il sistema si predispone in kernel mode.

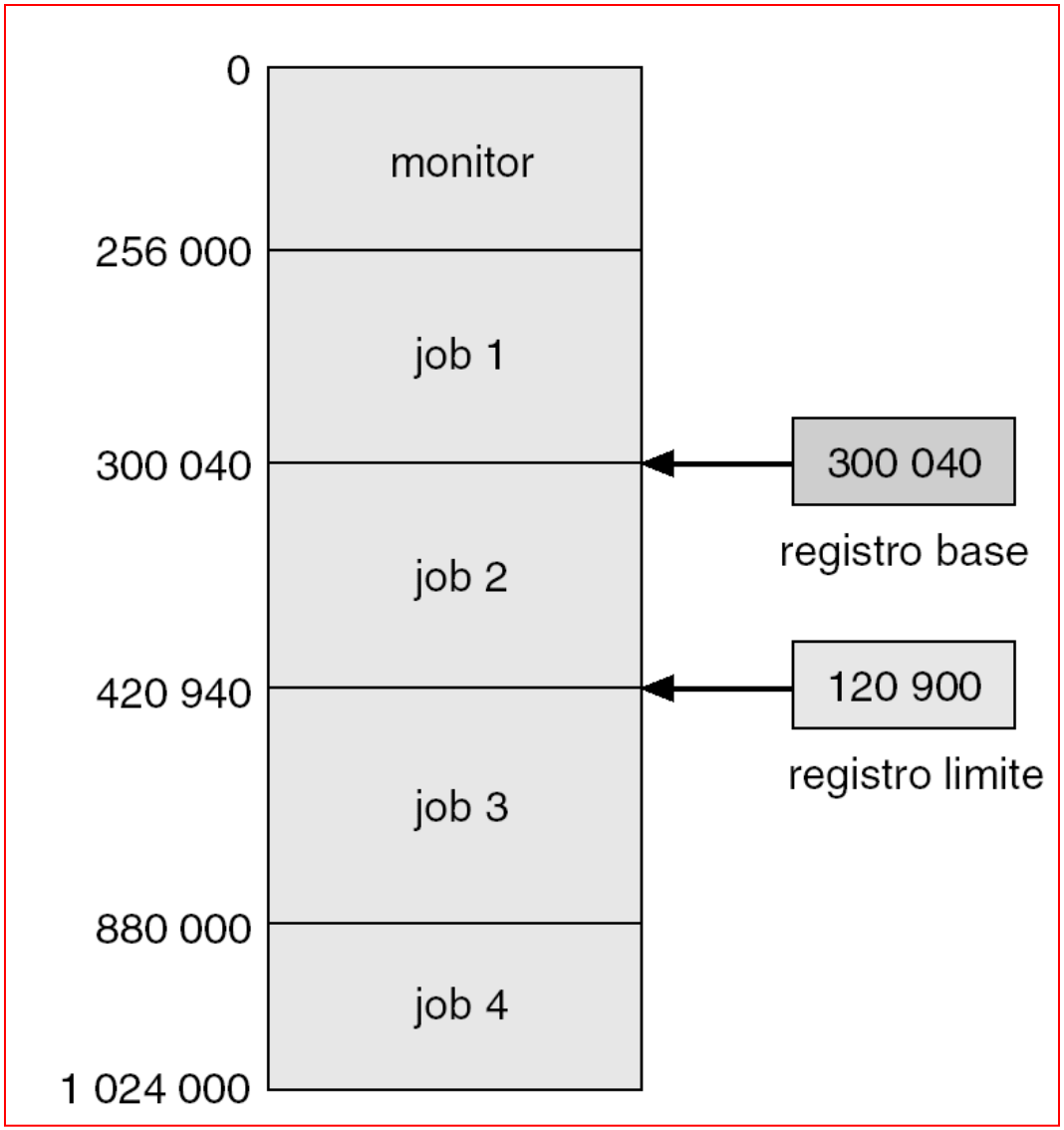
Protezione dell'I/O

- Per evitare che i processi utente effettuino I/O non consentiti, tutte le istruzioni di I/O sono privilegiate.
- I processi utente non potranno che eseguire operazioni di I/O attraverso System call
- Affinchè la protezione I/O sia completa un programma utente non deve mai poter ottenere il controllo della macchina in modalità supervisore.
  - In caso contrario il programma potrebbe commutare in modalità supervisore tutte le volte che si presenta un interrupt o una trap, saltando all'indirizzo determinato dal vettore di interrupt.
  - In tal caso un programma utente malizioso potrebbe memorizzare (come parte della sua esecuzione) un nuovo indirizzo per la routine di gestione dell'interruzione nell'interrupt vector.
  - In tal modo al successivo interrupt il programma potrebbe progressivamente alterare l'intero vettore di interruzioni e carpire il controllo della macchina in modalità supervisore.

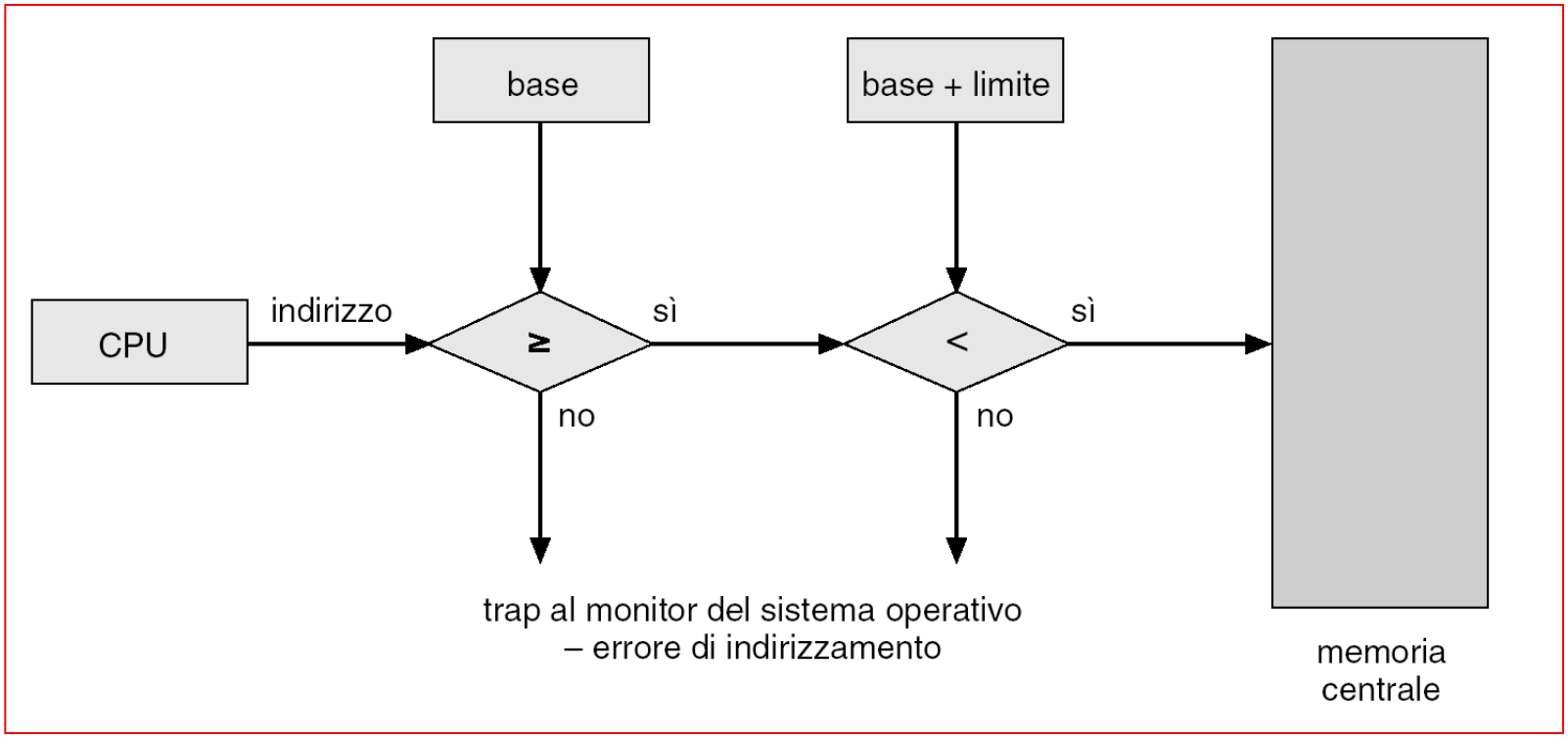
Protezione della memoria

- È necessario assicurare la protezione della memoria quanto meno per il vettore di interrupt e le routine di gestione degli interrupt del sistema operativo.
- Per avere protezione della memoria, vengono aggiunti due registri che determinano il range degli indirizzi fisici a cui un programma può accedere:
  - **Registro base** – contiene il più piccolo indirizzo legale della memoria fisica.
  - **Registro limite** – contiene la dimensione della gamma degli indirizzi.
- La memoria al di fuori del range definito è protetta.
- L'HW di CPU confronta ogni indirizzo generato in modalità utente con il contenuto dei registri.
- I registri base/limite possono essere aggiornati solo dal S.O. mediante un'istruzione privilegiata speciale.

Registri base/limite



Protezione dell'indirizzo hardware



Protezione della CPU

- Un programma utente potrebbe bloccare la CPU in un ciclo infinito. Allo scopo si utilizza un *timer* (fisso o variabile).
- Il *timer* interrompe l'elaborazione dopo un periodo di tempo specificato per far sì che il sistema operativo mantenga il controllo.
  - Il contatore viene decrementato ad ogni ciclo di clock.
  - Quando il contatore raggiunge lo 0, genera un interrupt.
- All'azzerarsi del timer, il S.O. tratta l'interruzione valutando la possibilità di generare un fatal error o estendere il time slice riservato ad un processo utente.
- L'uso più comune del temporizzatore è quello che permette di realizzare il time-sharing. Al termine di ogni slice scandita dal timer il S.O. esegue il cambio di contesto computazionale.
- Un ulteriore uso del temporizzatore consiste nel calcolare il tempo corrente rispetto ad un dato istante iniziale.
- Il caricamento del temporizzatore è un'istruzione privilegiata.