



Capitolo 1: Introduzione

- Che cosa fa un Sistema Operativo
- Organizzazione di un sistema elaborativo
- Architettura degli elaboratori
- Struttura del sistema operativo
- Attività del sistema operativo
- Gestione dei processi
- Gestione della memoria
- Gestione della memoria di massa
- Protezione e sicurezza
- Strutture dati del kernel
- Ambienti di elaborazione
- Sistemi operativi open source





Cos'è un Sistema Operativo?

- Un programma che agisce come tramite tra l'utente e gli elementi fisici del calcolatore.
- E' un insieme di programmi (software) che:
 - ◆ gestisce gli elementi fisici di un calcolatore (hardware),
 - ◆ fornisce una piattaforma ai programmi di applicazione
 - ◆ agisce da intermediario tra l'utente e la struttura fisica del calcolatore





Cos'è un Sistema Operativo? (II)

■ Scopi di un sistema operativo:

- ◆ Eseguire programmi utente e rendere più semplice la soluzione dei problemi dell'utente.
- ◆ Rendere conveniente ed efficiente l'utilizzo del sistema di calcolo.

■ Un sistema operativo deve assicurare il corretto funzionamento di un calcolatore.

■ Funzioni del Sistema Operativo

- ◆ Estendere e astrarre l'hardware (per semplificare la programmazione, per rendere i programmi portabili, etc..).
 - ✓ (ad es. un "file" è un astrazione)
- ◆ Gestire le risorse
 - ✓ (ad es. suddividere stampanti, dischi, tempo di CPU fra più programmi)



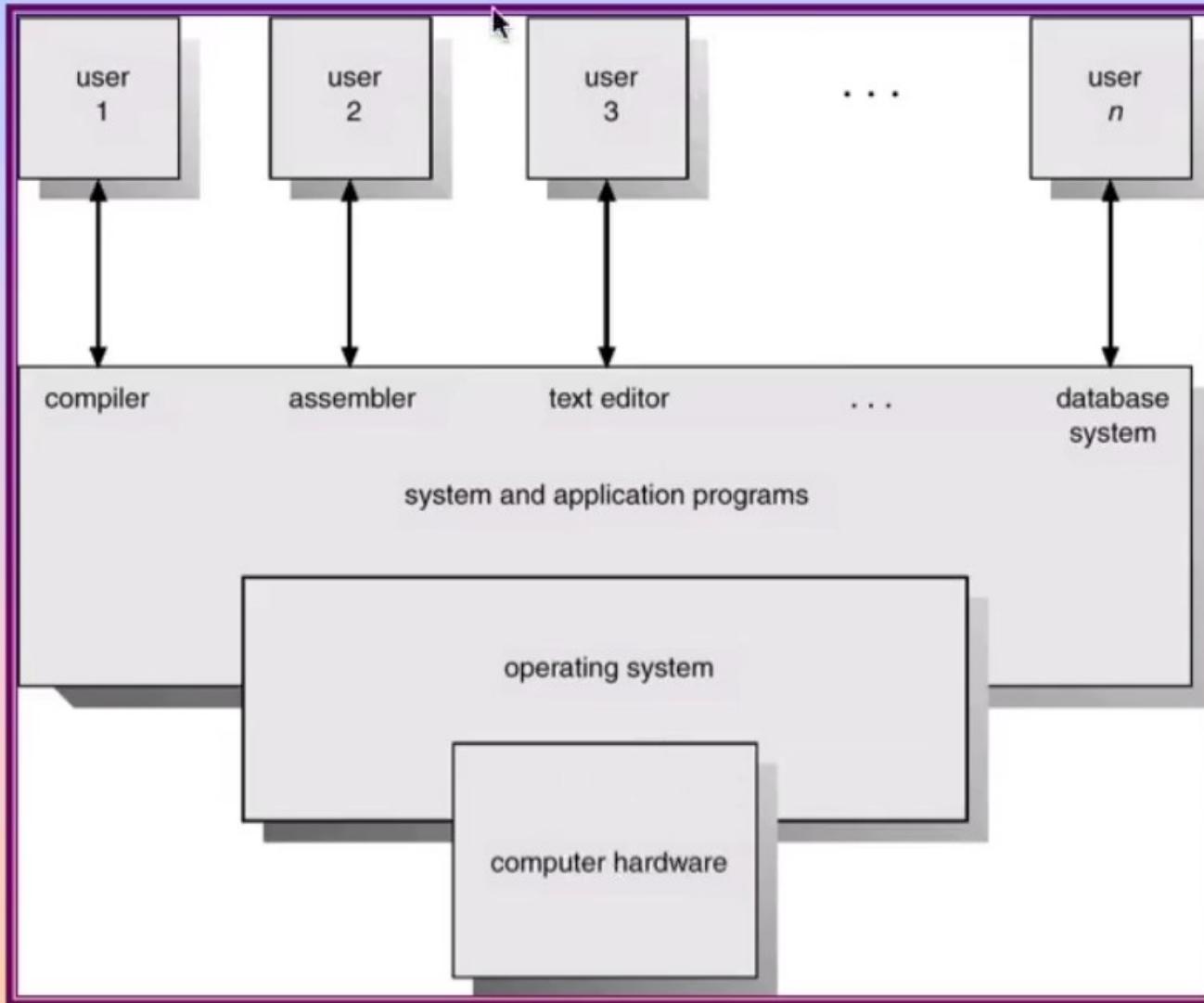


Componenti di un sistema di elaborazione

1. Hardware – (CPU, memoria, dispositivi di I/O, etc.).
2. Sistema Operativo – controlla e coordina l'uso delle risorse hardware su richiesta dei (vari) programmi applicativi dei (vari) utenti
3. Programmi applicativi – definiscono il modo in cui le risorse del sistema sono utilizzate per risolvere i problemi computazionali degli utenti (compilatori, database, video games, programmi finanziari, etc.).
4. Utenti (persone, macchinari, altri computer, etc.).



Componenti di un sistema di elaborazione (II)





Ruolo del Sistema Operativo

- I sistemi operativi (S.O. in breve) esistono perché forniscono agli utenti uno strumento conveniente per l'uso di un sistema di calcolo
- Convenienza:
 - ◆ facilità d'uso,
 - ◆ efficienza uso risorse.
- Gran parte della teoria dei S.O. si è concentrata sull'efficienza.
- Inoltre, hardware e S.O. si sono influenzati vicendevolmente.





Ruolo del Sistema Operativo: punto di vista dell'utente

■ PC

- ♦ Il sistema operativo è progettato principalmente per facilitare l'uso del computer.

■ Mainframe e Minicomputer

- ♦ Occorre massimizzare l'uso delle risorse.

■ Workstation

- ♦ Compromesso ottimale tra l'uso delle risorse individuali e risorse condivise.

■ Palmari e simili

- ♦ Progettati per l'uso individuale prestando attenzione alle prestazioni della batteria

■ Sistemi Embedded

- ♦ Concepiti per funzionare senza l'intervento dell'utente





Ruolo del Sistema Operativo: punto di vista del sistema

- Il sistema operativo è il programma più strettamente connesso con l'hardware.
- Quindi, è:
 - ◆ **allocatore di risorse:** di fronte a richieste conflittuali decide come assegnare equamente ed efficientemente le risorse ai programmi,
 - ◆ **programma di controllo:** garantisce l'esecuzione dei programmi senza errori e usi impropri del computer,
 - ◆ **esecutore di funzioni comuni:** esegue funzioni di utilità generale comuni ai diversi programmi (ad es. routine di I/O),
 - ◆ **nucleo (Kernel):** l'unico programma sempre in esecuzione (tutti gli altri sono "programmi applicativi").

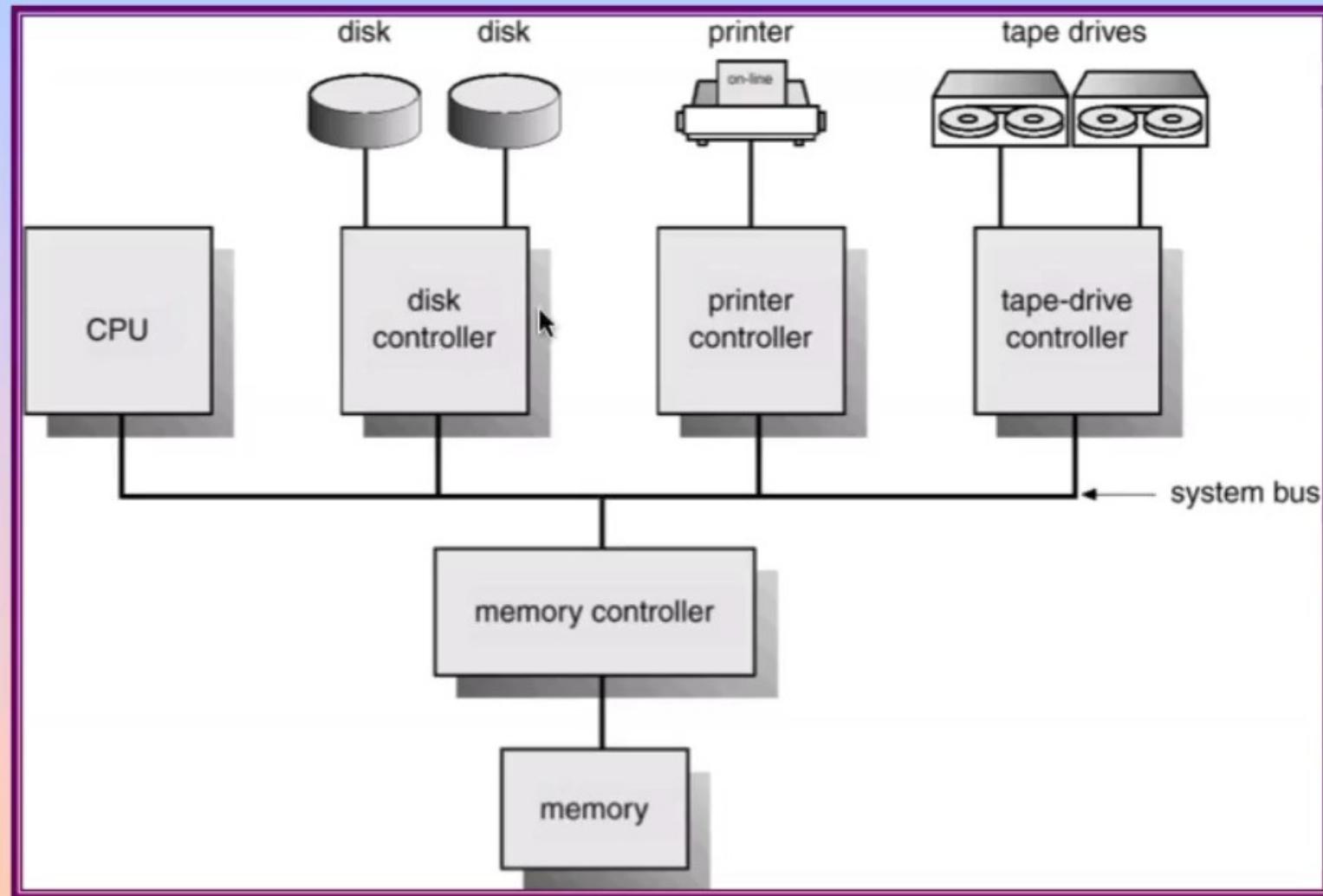




Sistemi di calcolo

- Al fine di evitare che i programmi utenti interferiscano con le operazioni proprie del sistema:
 - ◆ l'architettura del calcolatore deve fornire meccanismi appropriati per assicurarne il corretto comportamento e supportare tramite essi il S.O..
 - Un moderno calcolatore è composto da una CPU e da un certo numero di controllori di dispositivi connessi attraverso un canale di comunicazione comune (*bus*).
 - La sincronizzazione degli accessi alla memoria è garantita dalla presenza di un *controllore di memoria*.
 - L'avviamento del sistema richiede la presenza di uno specifico programma iniziale:
 - ◆ *programma di avviamento* o *bootstrap*, in genere contenuto in una ROM.
 - Questo programma
 - ◆ inizializza i registri della CPU, i controllori dei dispositivi, la memoria, etc.,
 - ◆ carica in memoria il sistema operativo
 - ◆ ne avvia l'esecuzione.
- 

Un moderno sistema di calcolo





Funzionamento di un sistema di calcolo

- I dispositivi di I/O e la CPU possono operare in modo concorrente.
- Ciascun controllore di dispositivo è responsabile di un dispositivo di uno specifico tipo.
- Ciascun controllore di dispositivo ha un buffer locale.
- La CPU sposta dati dalla/alla memoria principale verso/da i buffer locali.
- L' I/O è tra il dispositivo ed il buffer locale del suo controllore.
- I controllori di dispositivo informano la CPU della fine di una operazione di I/O tramite un segnale di interruzione (*interrupt*).





Segnali di interruzione

- Un segnale di interruzione causa il trasferimento del controllo all'appropriata procedura di servizio dell'evento ad esso associato.
- La gestione di un'interruzione deve essere molto rapida.
- Dato che il numero di possibili interrupt è predefinito, per gestire le interruzioni si utilizza una tabella di puntatori (*vettore delle interruzioni*):
 - ◆ che contiene gli indirizzi delle procedure di servizio associate all'interruzione.
- E' inoltre necessario:
 - ◆ salvare l'indirizzo dell'istruzione che viene interrotta
 - ◆ Disabilitare la possibilità di ulteriori interruzioni fino a quando la gestione dell'interrupt corrente non è terminata
- Un *segnale di eccezione (trap)* è un interrupt software causato da un errore o da una richiesta specifica dell'utente
 - ◆ *chiamata del sistema o del supervisore*





Struttura della memoria

- Memoria centrale:

- ◆ *memoria ad accesso diretto* - RAM, detta anche principale o volatile.

- E' realizzata con una tecnologia basata sui semiconduttori detta memoria dinamica ad accesso diretto (DRAM)

- ◆ ed è strutturata come un vettore di parole di memoria.

- In teoria si vorrebbe che nella RAM risiedessero sia i programmi che i dati.

- Impossibile:

- ◆ La capacità della memoria centrale non è di solito sufficiente
 - ◆ La memoria centrale è *volatile*.

- Memoria secondaria:

- ◆ tramite cui si realizza una estensione non volatile della memoria centrale,
 - ◆ generalmente di grande capacità.
 - ◆ dischi magnetici, dischi ottici e nastri magnetici.





Memoria centrale

- Gli unici dispositivi di memoria accessibili dalla CPU sono la memoria centrale ed i registri interni alla stessa CPU.
- Esistono istruzioni macchina che accettano indirizzi di memoria come argomenti:
 - ◆ nessuna istruzione accetta indirizzi di un disco.
- Tutte le istruzioni in esecuzione, unitamente ai dati da esse elaborati devono essere trasferiti in memoria prima che la CPU possa operare su di essi.





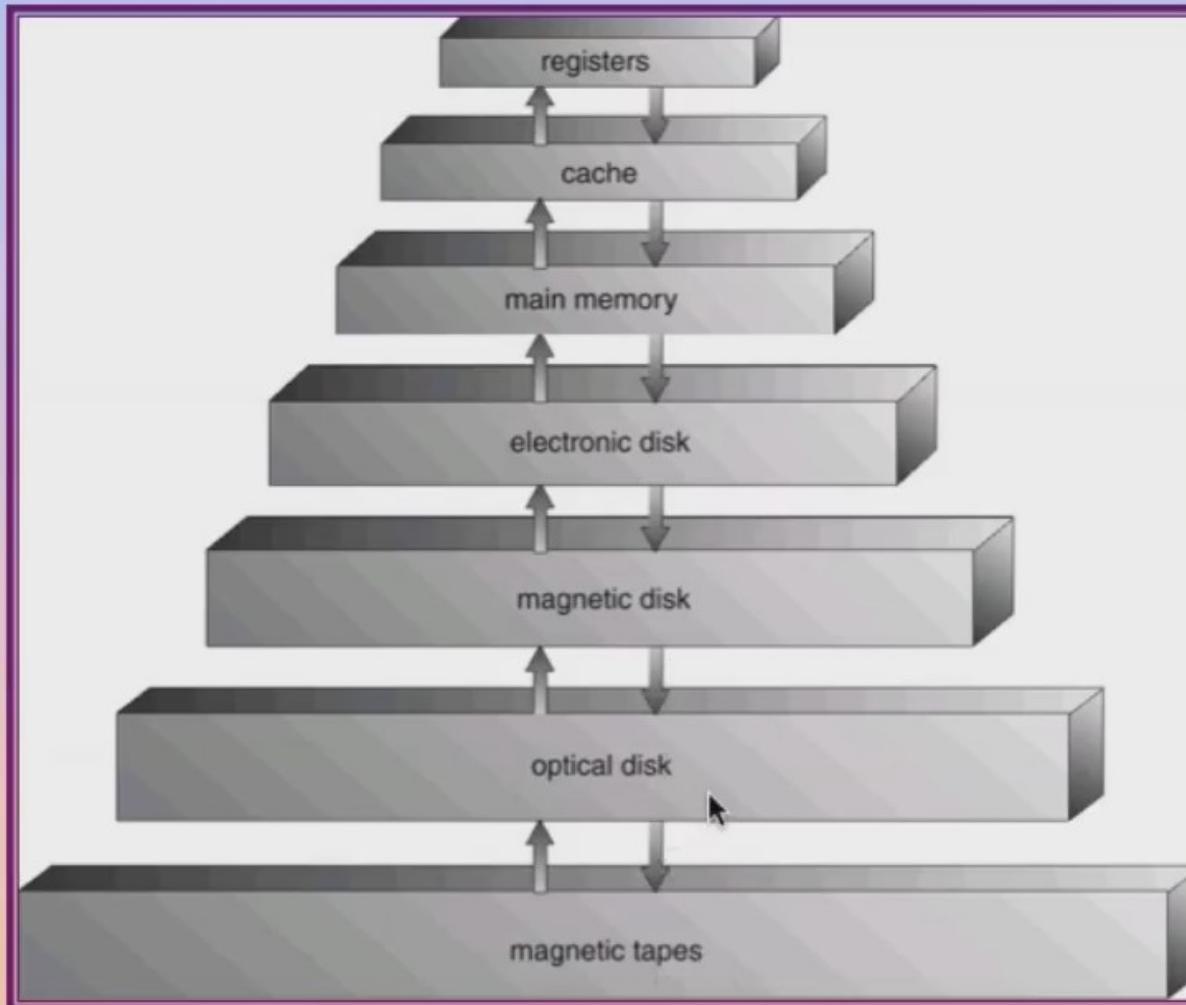
Gerarchia delle memorie

- I componenti di memoria di un sistema di calcolo si possono organizzare una struttura gerarchica, tenendo conto di:
 - ◆ velocità
 - ◆ costo
 - ◆ volatilità
- La memoria *cache* è una unità di memoria più veloce della memoria centrale.
- Il *caching* è il copiare informazioni in sistemi di memoria più veloci.
- La memoria centrale può essere vista come una *cache* per la memoria secondaria.





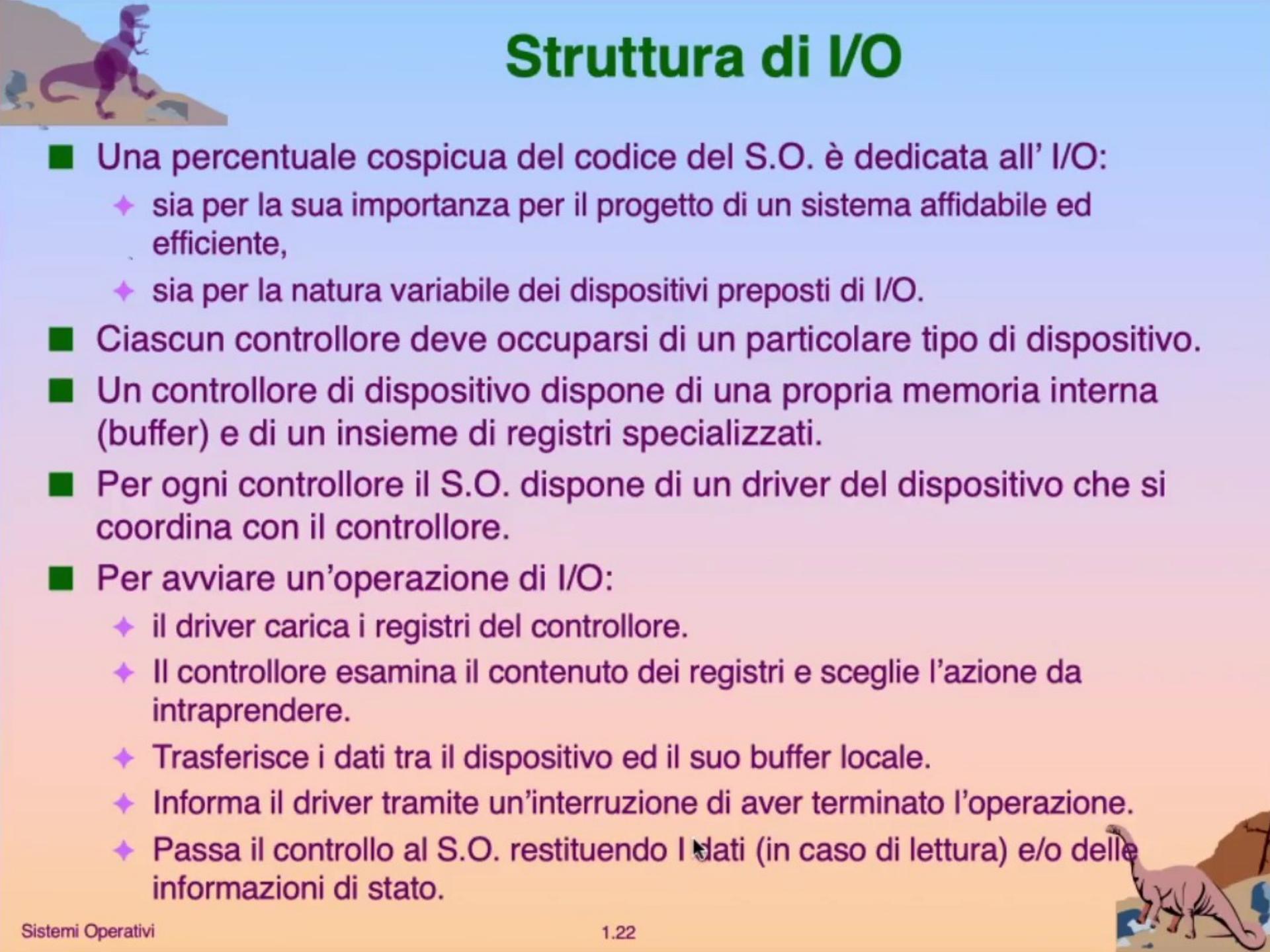
Gerarchia dei dispositivi di memoria





Caching

- Utilizzo di memoria ad alta velocità per memorizzare dati che sono stati recentemente utilizzati.
 - Quando si deve accedere ad una informazione si controlla se è presente all'interno della cache
 - ◆ altrimenti la si preleva dalla memoria centrale e la si copia nella cache.
 - Richiede una politica di *gestione della cache (cache management policy)*.
 - Il livello di caching nella gerarchia dei dispositivi di memoria implica che alcuni dati sono memorizzati simultaneamente in più di un livello.
 - Nascono quindi, soprattutto in sistemi time sharing o con più unità di elaborazione, problemi di *consistenza dei dati* e *coerenza della cache*.
- 



Struttura di I/O

- Una percentuale cospicua del codice del S.O. è dedicata all' I/O:
 - ◆ sia per la sua importanza per il progetto di un sistema affidabile ed efficiente,
 - ◆ sia per la natura variabile dei dispositivi preposti di I/O.
- Ciascun controllore deve occuparsi di un particolare tipo di dispositivo.
- Un controllore di dispositivo dispone di una propria memoria interna (buffer) e di un insieme di registri specializzati.
- Per ogni controllore il S.O. dispone di un driver del dispositivo che si coordina con il controllore.
- Per avviare un'operazione di I/O:
 - ◆ il driver carica i registri del controllore.
 - ◆ Il controllore esamina il contenuto dei registri e sceglie l'azione da intraprendere.
 - ◆ Trasferisce i dati tra il dispositivo ed il suo buffer locale.
 - ◆ Informa il driver tramite un'interruzione di aver terminato l'operazione.
 - ◆ Passa il controllo al S.O. restituendo i dati (in caso di lettura) e/o delle informazioni di stato.



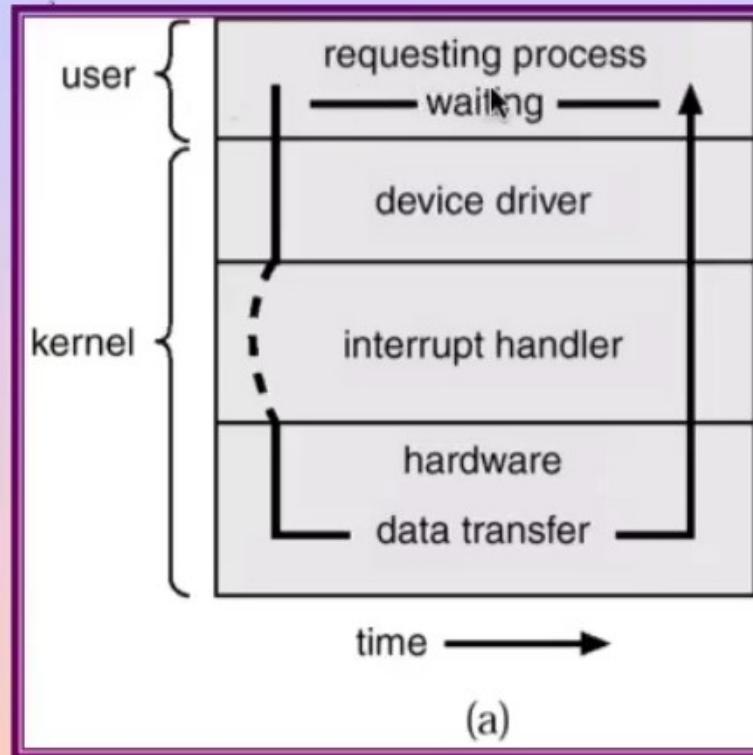
I/O sincrono e asincrono

- Una volta iniziata un'operazione di I/O, il corso dell'esecuzione può seguire due percorsi distinti:
- I/O *sincrono*:
 - ◆ dopo l'inizio dell' I/O il controllo ritorna al programma utente solo dopo il completamento dell'operazione di I/O.
 - ◆ Può essere servita al più una richiesta di I/O per volta non e' cioé possibile servire più richieste di I/O simultaneamente.
- I/O *asincrono*:
 - ◆ dopo l'inizio dell' I/O il controllo ritorna al S.O. o ad un programma utente senza attendere il completamento dell'operazione di I/O.
- Due possibilità per fermare il programma utente ove richiesto:
 - ◆ Ciclo di attesa (busy waiting).
 - ◆ Una chiamata a sistema di *wait* rende la CPU inattiva fino alla prossima interruzione che segnala la fine dell'I/O.

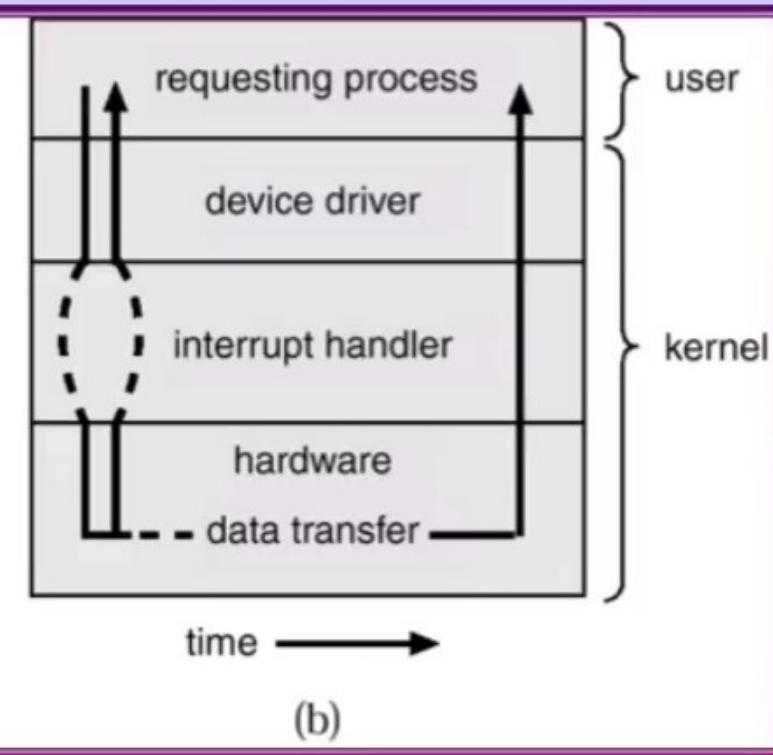


Due metodi di I/O

Sincrono



Asincrono





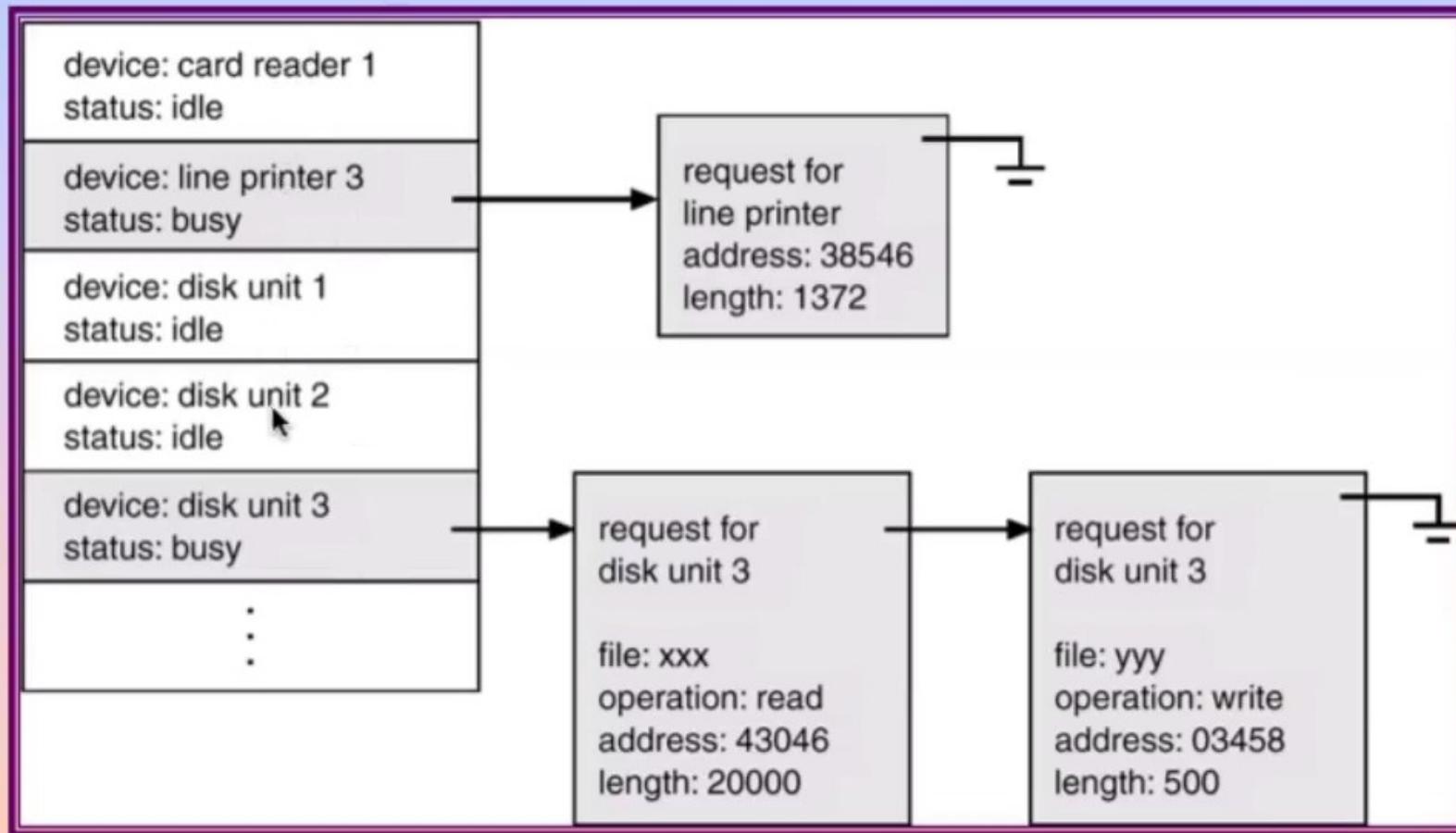
I/O sincrono e asincrono (II)

- Il sistema deve tener traccia delle varie richieste di I/O attive allo stesso istante tramite una *tabella di stato dei dispositivi*.
- Questa contiene un record per ciascun dispositivo di I/O indicante:
 - ◆ il tipo di dispositivo a cui fa riferimento,
 - ◆ il suo indirizzo e
 - ◆ il suo stato (disattivato, inattivo, occupato).
- Se un controllore di dispositivo di I/O invia un'interruzione per richiedere un servizio,
 - ◆ S.O. individua il controllore ed accede alla tabella dei dispositivi,
 - ◆ risale allo stato del dispositivo,
 - ◆ modifica l'elemento della tabella indicando l'occorrenza di un interrupt.
- Al completamento della richiesta, se un processo attendeva il temine dell'I/O (come risulta dalla tabella), è possibile restituirgli il controllo.





Tabella di stato dei dispositivi





Accesso diretto alla memoria (DMA)

- Utilizzato per permettere a dispositivi di I/O molto veloci di trasmettere informazioni ad una velocità vicina a quella della memoria.
- I controllori di dispositivo trasferiscono blocchi di dati direttamente dal buffer del controllore nella memoria principale senza alcun intervento della CPU.
- In questo modo il trasferimento richiede una sola interruzione per blocco di dati trasferito, piuttosto che per ogni parola come avviene nella gestione dei dispositivi più lenti.
- Il sistema individua l'area di memoria interessata al DMA (in genere da 128 a 4096 byte).
- Il *driver di dispositivo* imposta i registri del controllore di DMA, che riceve quindi l'istruzione di avvio dell'operazione di I/O.
- Mentre il controllore DMA esegue il trasferimento dei dati, la CPU è libera di eseguire altri compiti.
- Il controllore DMA notifica alla CPU il completamento dell'I/O tramite interrupt.





Architetture

- Sistemi monoprocessoress.
 - Sistemi multiprocessoress.
 - Cluster di elaboratori.
- 

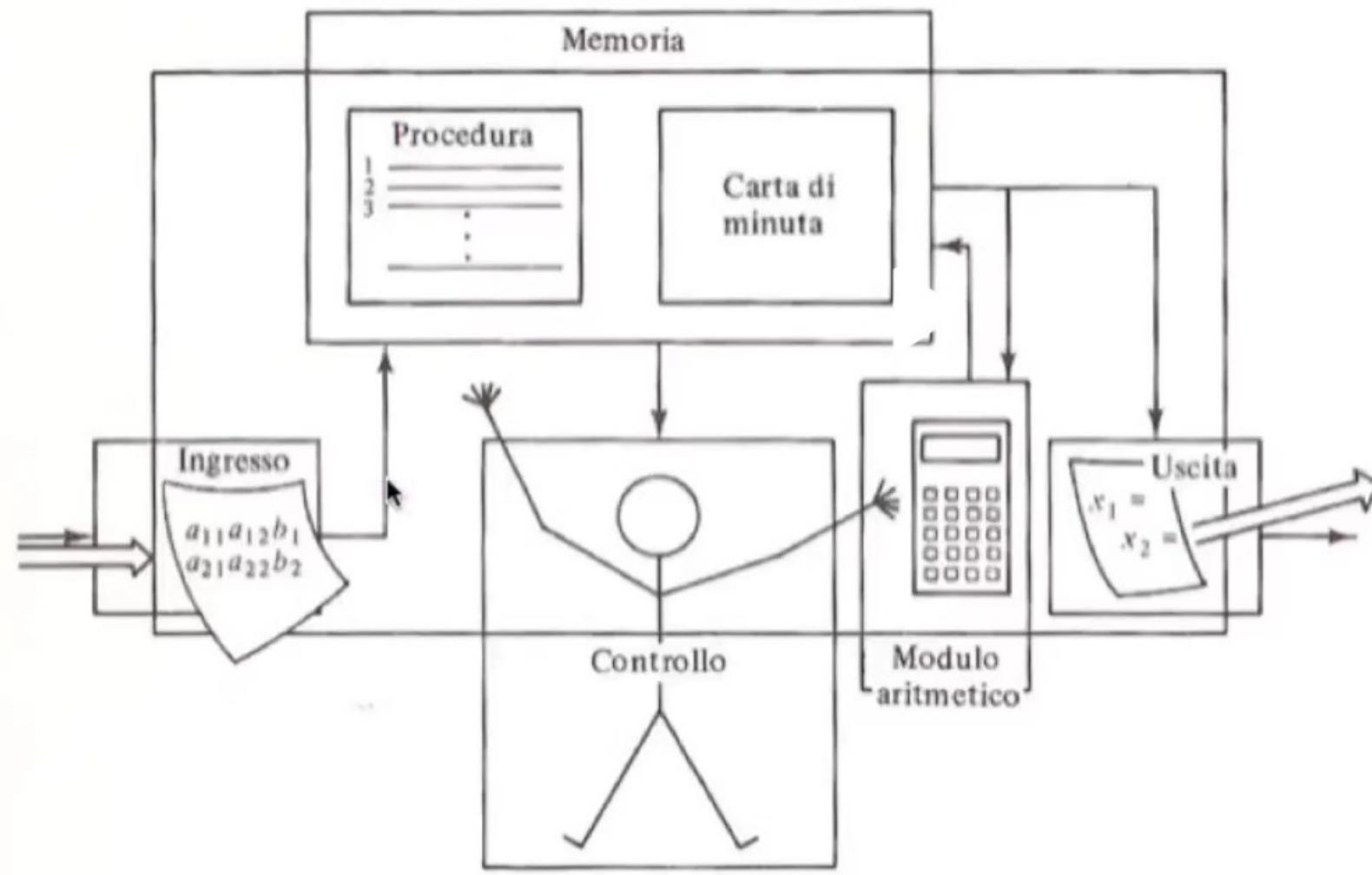


Architetture a singolo processore

- Questi sistemi sono dotati di un singolo processore che esegue un set di istruzioni general-purpose.
- Spesso usano anche processori special purpose,
 - ◆ ad es. i processori di I/O, che muovono velocemente dati tra le componenti (disk-controller, processori associati alle tastiere, etc.).
- A volte la CPU principale comunica con questi processori.
- Altre volte, essi sono totalmente autonomi.



Architettura di Von Neumann





Sistemi multiprocessore

- Sistemi dotati di più unità di elaborazione.
 - ◆ Sistemi paralleli detti anche sistemi strettamente connessi - tightly coupled systems.
- I processori condividono la memoria ed un clock; comunicano generalmente tramite la memoria condivisa.
- Vantaggi:
 - ◆ Maggiore produttività (*throughput*):
 - ✓ svolgere un lavoro maggiore in minor tempo.
 - ◆ Economia di scala:
 - ✓ condividere periferiche, alimentatori, etc. = risparmio.
 - ◆ Incremento dell' affidabilità:
 - ✓ un guasto non blocca il sistema ma lo rallenta.
 - ✓ Degradazione controllata (graceful degradation) e sistemi *fault-tolerant*.



Struttura del sistema operativo

- Concetto chiave è quello della **multiprogrammazione**:
 - ◆ necessaria per aumentare l'efficienza.
- Un solo utente non può tenere CPU e dispositivi I/O occupati per tutto il tempo.
- La multiprogrammazione consente di aumentare la percentuale di utilizzo della CPU organizzando i lavori in modo tale da mantenerla in continua attività.
- Un sottoinsieme dei job si trova in memoria centrale (**job pool**).
- Un job viene selezionato (**job scheduling**) ed eseguito.
- Quando il job è in attesa (ad es. di un'operazione di I/O), il S.O. esegue un altro job.





Struttura del sistema operativo (II)

- Altro concetto chiave: **timesharing** (multitasking):
 - ↳ estensione logica della multiprogrammazione,
 - ↳ la CPU commuta tra i job così frequentemente che gli utenti possono interagire con ciascun job mentre è in esecuzione, realizzando una computazione interattiva.
- Tempo di Risposta < 1 secondo.
- Ciascun utente ha almeno un processo in esecuzione in memoria.
- Se diversi processi sono pronti per essere eseguiti sarà necessaria la **schedulazione** della CPU.
- Se lo spazio di memoria non è sufficiente per contenere tutti i processi,
 - ↳ tramite lo swapping alcuni processi verranno spostati temporaneamente su memoria di massa e poi riportati in memoria centrale per essere eseguiti.
- La **memoria virtuale** permette l'esecuzione di processi che non sono completamente in memoria e separa la memoria fisica da quella logica.





Attività del S.O.: gestione delle interruzioni

- I sistemi operativi moderni sono caratterizzati dal fatto di essere *guidati dalle interruzioni (interrupt driven)*.
 - ◆ se non ci sono processi da eseguire, dispositivi di I/O da servire o utenti con cui interagire, il S.O. resta inattivo nell'attesa che accada qualcosa.
- In presenza di una interruzione:
 - ◆ il sistema operativo preserva lo stato della CPU salvando lo stato dei registri e del contatore di programma prima di servire l'interruzione.
 - ◆ Determina di che tipo sia l'interruzione.
 - ◆ Segmenti diversi di codice determinano quale azione debba essere presa per ciascun tipo di interrupt.
 - ◆ Dopo aver servito l'interruzione il S.O. ripristina lo stato della CPU (ad es. i registri) e del contatore di programma originali.



Duplice modo di funzionamento (dual mode)

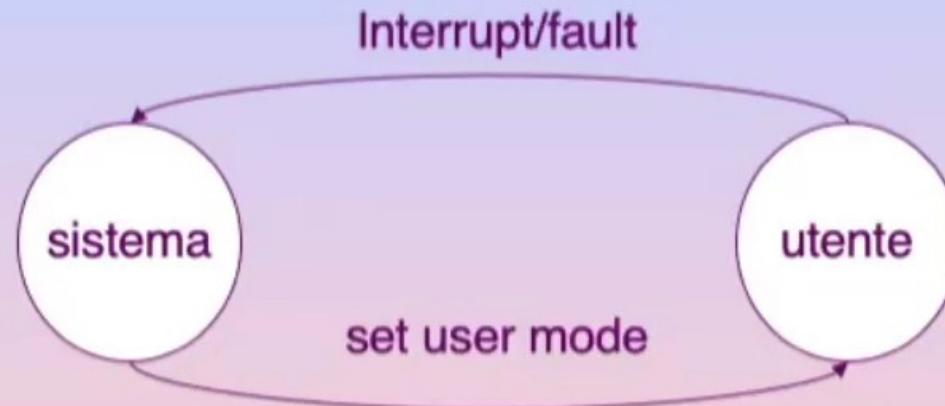
- La protezione deve essere garantita per qualsiasi risorsa condivisibile del sistema.
- L'architettura del sistema deve supportare almeno due distinti *modi* di funzionamento:
 1. *Modo d'utente (User mode)* in cui avviene l'esecuzione dei programmi utente
 2. *Modo di sistema (Monitor mode o kernel mode o system mode)* in cui avviene l'esecuzione delle chiamate e dei programmi di sistema.
- Un *bit di modo (mode bit)* di cui deve essere dotata l'architettura (hardware) della CPU indica il modo corrente:
 - ◆ sistema (0),
 - ◆ utente (1).





Duplice modo di funzionamento (dual mode) (II)

- In presenza di un'interruzione o eccezione l'hardware commuta il modo di sistema.



- Le *istruzioni privilegiate* possono essere date solo in modo di sistema
- L'utente, per richiedere un servizio al Sistema Operativo, utilizza una *chiamata di funzione del sistema operativo*, detta anche *chiamata del sistema* (*system call*),
 - ◆ gestita dal sistema tramite interrupt.





Gestione dei processi

- Un *processo di elaborazione* si può considerare come un “programma in esecuzione”.
- Un programma da per se non è un processo:
 - ❖ un programma è un’entità passiva, come il contenuto di un file memorizzato su disco,
 - ❖ mentre un processo è un’entità attiva, con un contatore di programma.
- Un processo necessita di alcune risorse, tra cui tempo di CPU, memoria, accesso ai files e ai dispositivi di I/O.
- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei processi:
 - ❖ Creazione e cancellazione dei processi utenti e di sistema.
 - ❖ Sospensione e ripristino dei processi.
 - ❖ Fornitura di meccanismi per:
 - ✓ Sincronizzazione dei processi
 - ✓ Comunicazione tra processi
 - ✓ Gestione delle situazioni di stallo (*deadlock*)





Gestione della memoria centrale



- La memoria è un vasto vettore di dimensioni che variano tra le centinaia di migliaia ed i miliardi di parole
- E' un "magazzino" di dati velocemente accessibili condivisi dalla CPU e da alcuni dispositivi di I/O.
- La memoria centrale contiene memorie "volatili", che perdono il loro contenuto in caso di mancanza di alimentazione.
- Il S.O. è responsabile delle seguenti attività connesse alla gestione della memoria centrale:
 - ◆ Tenere traccia di quali parti della memoria sono attualmente usate e da che cosa.
 - ◆ Decidere quali processi si debbano caricare nella memoria quando vi sia spazio disponibile.
 - ◆ Assegnare e revocare lo spazio di memoria secondo le necessità.





Gestione dei file



- Un file è una raccolta di informazioni correlate definite dal loro creatore.
- Comunemente i file rappresentano programmi (codice sorgente o oggetto) e dati.
- S.O. fornisce una visione logica uniforme del processo di registrazione delle informazioni:
 - ◆ astrae le caratteristiche fisiche dei dispositivi per definire una unità di memorizzazione, cioè il *file*.
- S.O. associa i file ai mezzi fisici e vi accede attraverso i dispositivi che li controllano.
- S.O. è responsabile delle seguenti attività connesse alla gestione dei file:
 - ◆ Creazione e cancellazione di file.
 - ◆ Creazione e cancellazione di directory.
 - ◆ Fornitura delle funzioni fondamentali per la gestione di file e directory.
 - ◆ Associazione dei file ai dispositivi di memoria secondaria.
 - ◆ Creazione di copie di riserva (backup) dei file su dispositivi di memorizzazione non volatili.



Gestione del sistema di I/O

→

- Uno tra gli scopi di un S.O. è nascondere all'utente le caratteristiche degli specifici dispositivi.
- Un *sottosistema di I/O* consiste delle parti seguenti:
 - ◆ Un componente di gestione della memoria comprendente
 - ✓ la gestione delle regioni della memoria riservate ai trasferimenti di I/O (*buffer*),
 - ✓ la gestione della cache
 - ✓ la gestione asincrona delle operazioni di I/O e dell'esecuzione di più processi (*spooling*).
 - ◆ Un'interfaccia generale per i driver dei dispositivi.
 - ◆ I driver per gli specifici dispositivi.





Gestione della memoria di massa



- Giacché la memoria centrale è volatile ed è troppo piccola per contenere tutti i dati e tutti i programmi permanentemente,
 - ◆ il calcolatore deve disporre di una memoria secondaria, non volatile, in ausilio alla memoria centrale.
- I dischi sono uno dei mezzi più usati per la memorizzazione secondaria, su di essi vengono memorizzati sia dati che programmi.
- S.O. è responsabile delle seguenti attività connesse alla gestione dei dischi:
 - ◆ Gestione dello spazio libero
 - ◆ Assegnazione dello spazio
 - ◆ Scheduling del disco





Protezione dell'I/O



- Tutte le istruzioni di I/O sono istruzioni privilegiate.
- E' necessario evitare che l'utente possa in qualche modo ottenere il controllo del calcolatore quando questo è in modo di sistema,
 - ◆ ad esempio un utente non deve poter modificare il vettore delle interruzioni.



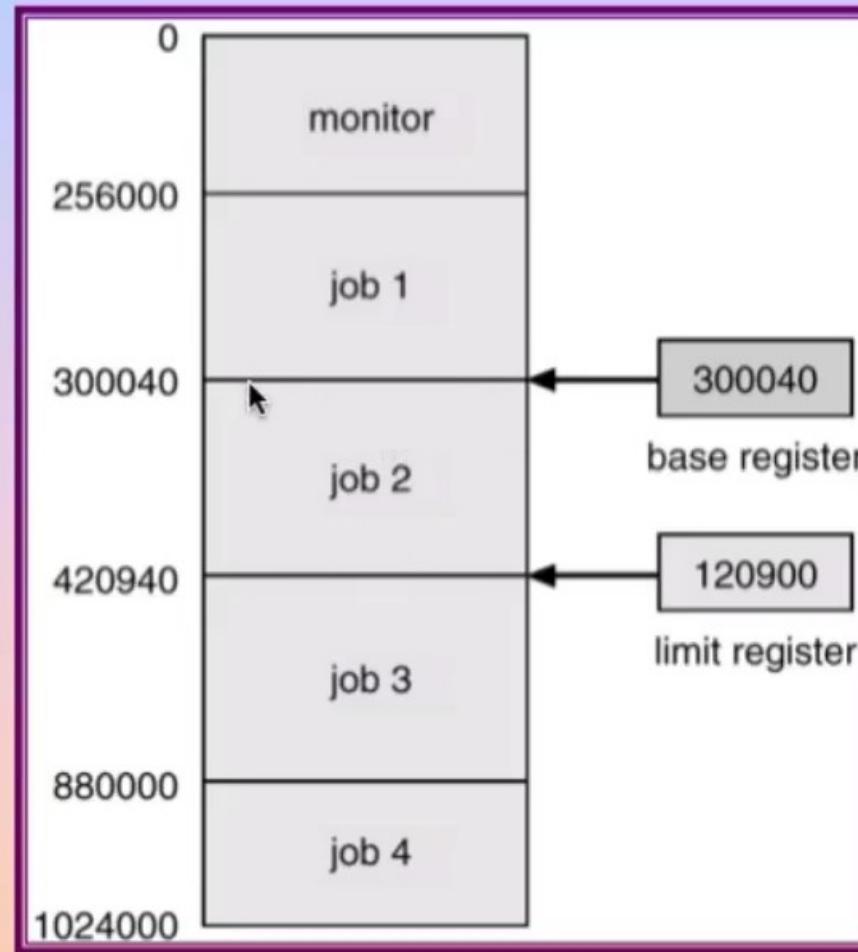
Protezione della memoria (I)

- Bisogna proteggere il vettore delle interruzioni e anche le procedure di servizio dei segnali di interruzione.
- Per separare lo spazio di memoria dei programmi serve la capacità di determinare l'intervallo di indirizzi cui il programma può accedere.
- Due registri determinano il range di indirizzi legali a cui un programma può accedere:
 - ◆ **Registro di base (base register)** – contiene il più basso indirizzo della memoria fisica al quale il programma può accedere
 - ◆ **Registro di limite (limit register)** – contiene la dimensione dell'intervallo.
- La memoria al di fuori dell'intervallo individuato dai due registri non deve essere accessibile al programma.





Uso di un registro di base e di un registro di limite



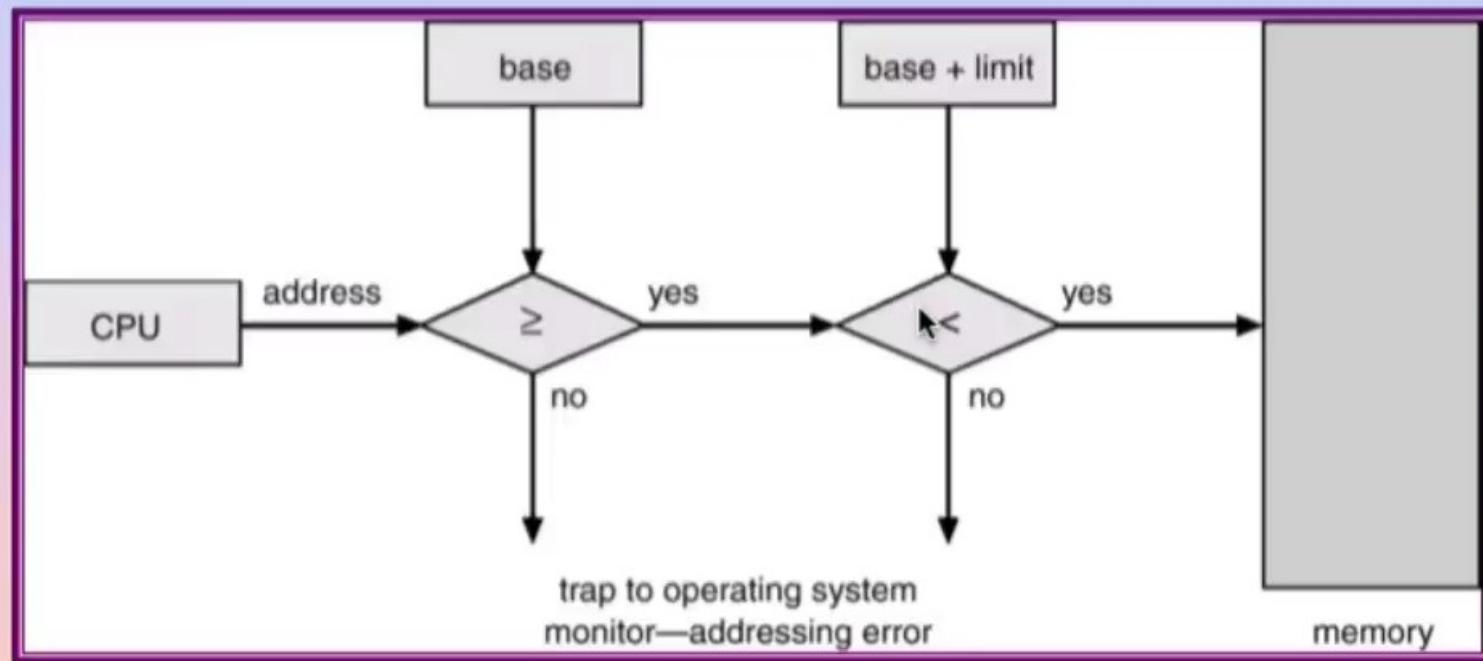


Protezione della memoria (II)

- Funzionando in modo di sistema S.O. può accedere sia alla memoria ad esso riservata sia a quella riservata agli utenti.
- Le istruzioni di caricamento dei registri di base e di limite devono essere istruzioni privilegiate.



Architettura di protezione degli indirizzi con registri di base e di limite





Protezione della CPU

- Occorre assicurare che S.O. mantenga il controllo dell'elaborazione,
 - ◆ cioè impedire che un programma utente entri in un ciclo infinito senza più restituire il controllo.
- *Temporizzatore* – interrompe l'esecuzione di un processo dopo un periodo predeterminato per assicurare che il S.O. mantenga il controllo della CPU.
 - ◆ Il temporizzatore è decrementato ad ogni ciclo di clock.
 - ◆ Quando il temporizzatore arriva a zero viene generato un interrupt.
- Il temporizzatore viene generalmente utilizzato soprattutto nei sistemi a partizione di tempo (time sharing).
- Può essere utilizzato anche per determinare l'ora corrente
- Il caricamento del temporizzatore è un'istruzione privilegiata.

