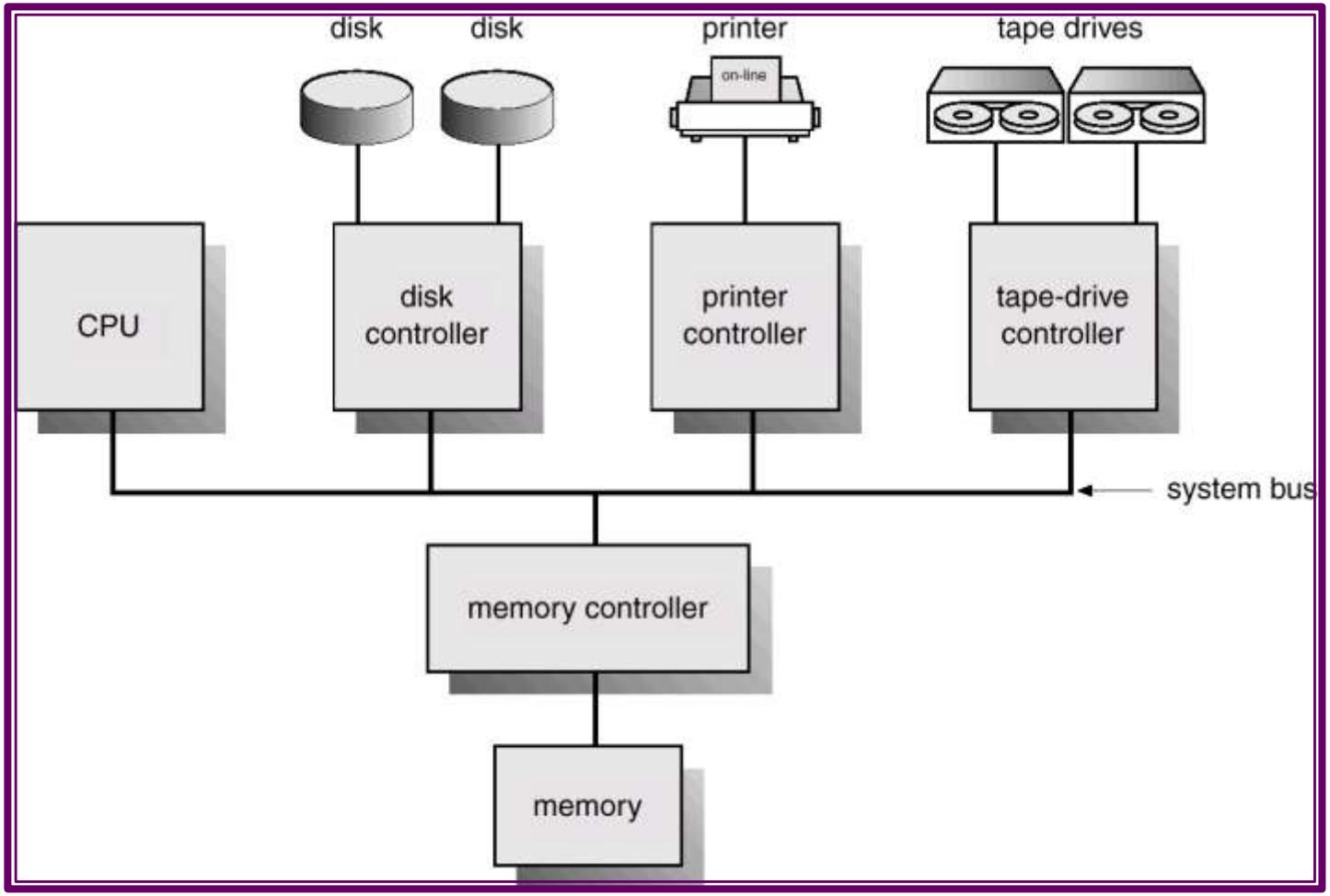
PROCESSOR AND O.S. MAIN FEATURES

A generic computer system architecture



PROCESSOR AND O.S. MAIN FEATURES

A generic computer system operation

The initial program (***bootstrap program***) stored in the ROM initializes the system (registers, controllers, memory).

Then localizes, locates and loads into memory the ***kernel***, starting the first process (***initiator***) and waiting for some ***event*** to occurr.

An event is usually signaled by an ***interrupt*** from either the hardware or the software.

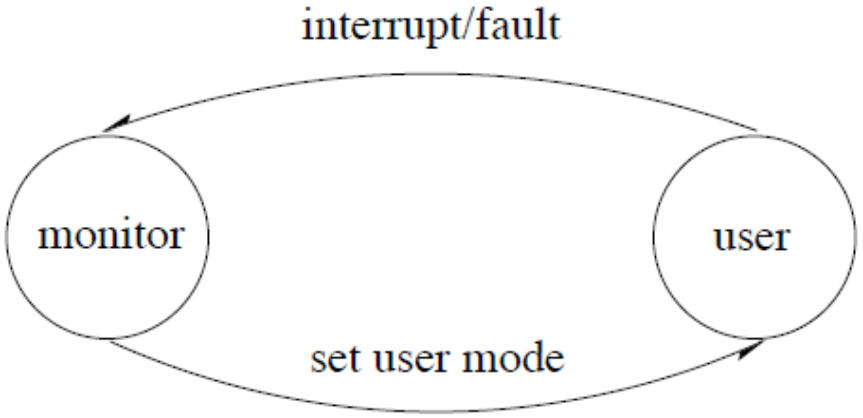
The O.S. is interrupt driven.

Hardware may trigger an interrupt by *sending a signal to the CPU*.
Software may trigger an interrupt by *executing a supervisor call* (SVC).

HARDWARE PROTECTION

Dual-Mode Operation

- Sharing system resources requires operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
- Provide hardware support to differentiate between at least two modes of operations.
  1. ***User mode*** – execution done on behalf of a user.
  2. ***Monitor mode*** (also *kernel mode* or *system mode*) – execution done on behalf of operating system.
- *Mode bit* added to computer hardware to show the current mode: monitor (0) or user (1).
- When an interrupt or fault occurs hardware switches to monitor mode.

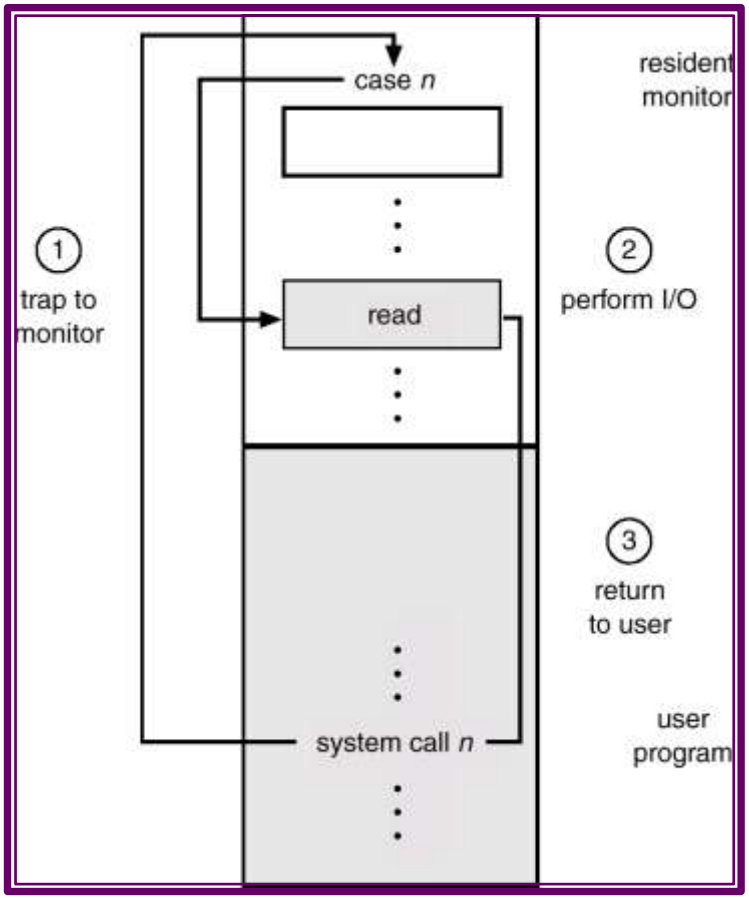


- *Privileged instructions* can be issued only in monitor mode.

HARDWARE PROTECTION

I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode (i.e., a user program that, as part of its execution, stores a new address in the interrupt vector).



PROCESSOR AND O.S. MAIN FEATURES

Tipi di interruzioni

Interruzione interna (o *trap*): è quella generata dal software ed è causata da un errore (divisione per zero, indirizzo errato di memoria) o dalla richiesta di un servizio del S.O. Ha carattere sincrono, in quanto il processo che lancia la trap va nello stato di wait provocando l'avvio di un segmento di codice del nucleo del sistema operativo (*Supervisor Call o SVC*).

Interruzione esterna (o *interrupt*): ha carattere asincrono, in quanto al verificarsi di un evento, la sua occorrenza è segnalata alla CPU tramite un segnale in ragione del quale viene provocato l'avvio di un segmento di codice del nucleo del sistema operativo.

Ogni processore ha una propria architettura di SVC ed interrupt, anche se le diverse architetture hanno molte caratteristiche in comune.

PROCESSOR AND O.S. MAIN FEATURES

SVC operation

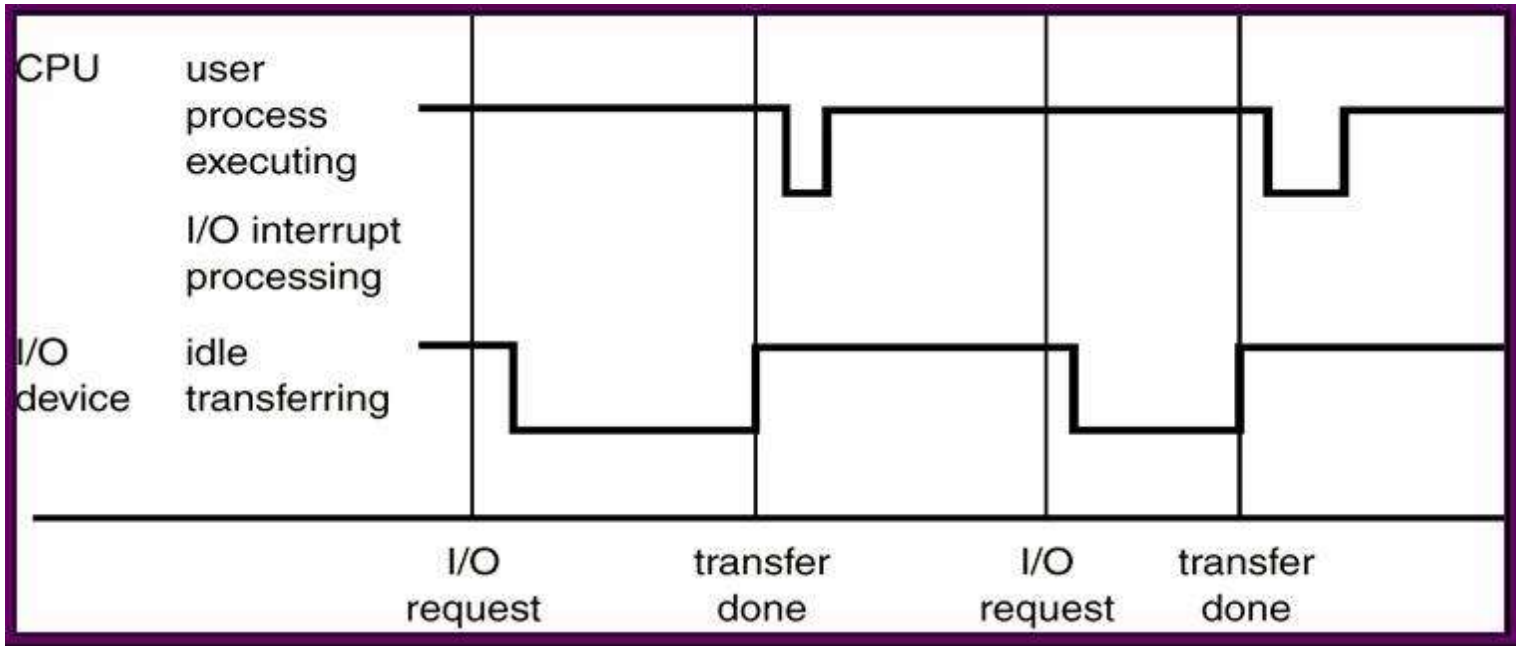
- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)

- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

PROCESSOR AND O.S. MAIN FEATURES

Interrupt operation

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are disabled (*interrupt masking*) while another interrupt is being processed to prevent a *lost interrupt*.



PROCESSOR AND O.S. MAIN FEATURES

Gestione di un interrupt

Ogni processore ha un proprio meccanismo di gestione degli interrupt.

Quando avviene un'interruzione il sistema operativo preserva lo stato della CPU.

Il meccanismo più semplice prevede che quando la CPU viene interrotta, si completi l'istruzione corrente, si salvi il contesto computazionale corrente (registri e program counter) in una locazione di memoria (fissa o indicizzata in base al dispositivo da cui è pervenuto l'interrupt) e si passi ad eseguire l'***interrupt handler***, che si trova ad un indirizzo prefissato di memoria e che, in base al codice dell'interrupt, trasferisce l'esecuzione all'indirizzo specificato nell'***interrupt vector***, che occupa le prime posizioni di memoria e che è costituito da tanti indirizzi quanti sono gli interrupt previsti dal processore. Ogni indirizzo rimanda al segmento di codice che serve quel particolare interrupt.

Per rendere più rapide le operazioni, negli attuali meccanismi di gestione degli interrupt viene usato direttamente il vettore delle interruzioni, senza interrupt handler.

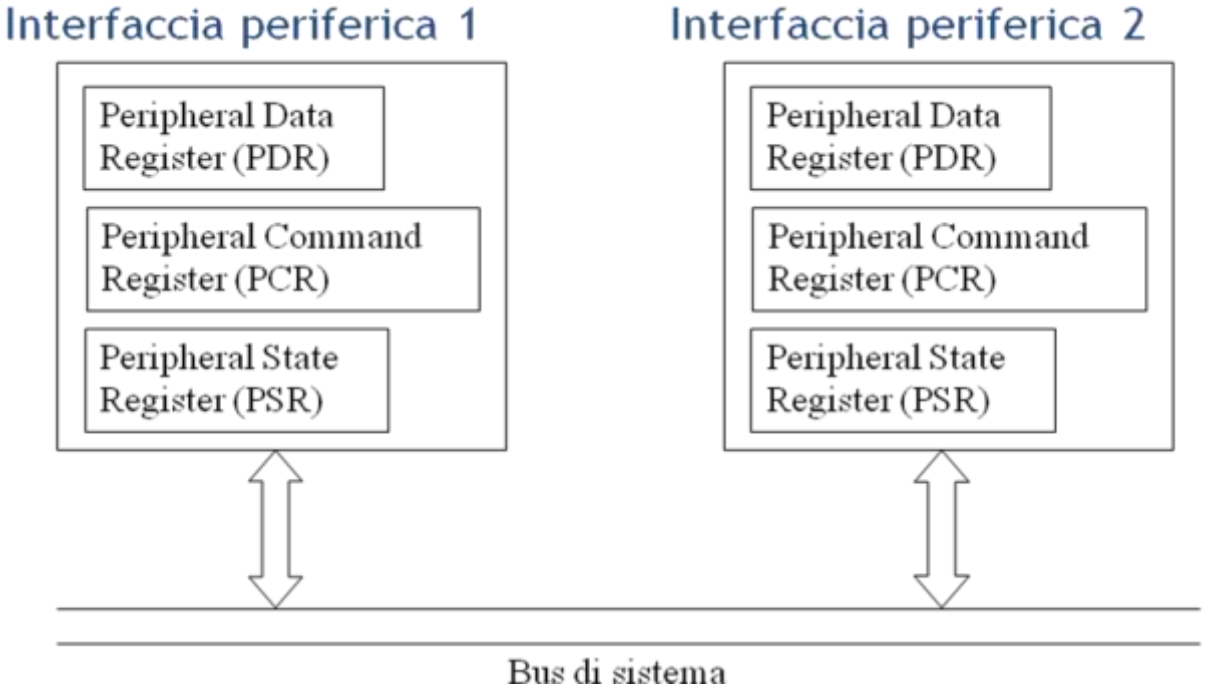
Al termine dell'esecuzione del segmento di codice indirizzato dall'interrupt vector, si ripristina il contesto computazionale e la CPU riprende l'esecuzione interrotta.

Nelle architetture più recenti il contesto computazionale viene salvato sullo stack del sistema.

Una SVC viene servita facendo anche essa riferimento al vettore delle interruzioni.

OPERAZIONI DEI SISTEMI DI CALCOLO

- I dispositivi di I/O e la CPU possono funzionare concorrentemente
- Ogni controller di dispositivo gestisce un particolare tipo di dispositivo.
- Ogni controller ha un buffer locale
- La CPU muove dati da/per la memoria principale per/da i buffer locali dei controller
- l'I/O avviene tra il dispositivo e il buffer locale del controller
- Il controller informa la CPU al termine della sua operazione, generando un interrupt.



PROCESSOR AND O.S. MAIN FEATURES

Servicing an I/O operation

To start an I/O operation the CPU loads the appropriate registers into the device controller.

The device controller examines the content of the registers to determin the action to take and starts the requested transfer of data. The I/O operation may be accomplished in two different ways: Synchronous or asynchronous I/O.

Once the transfer is completed, the controller sends an interrupt to the CPU to signal it has finished the requested operation.

PROCESSOR AND O.S. MAIN FEATURES

Invocazione del sistema operativo

Dato che le istruzioni di I/O sono privilegiate, come può il programma utente eseguire dell'I/O?

Attraverso le ***system call***, il metodo con cui un processo richiede un'azione da parte del sistema operativo:

- Solitamente sono un interrupt software (trap)
- Il controllo passa attraverso il vettore di interrupt alla routine di servizio della trap nel sistema operativo, e il mode bit viene impostato a "monitor".
- Il sistema operativo verifica che i parametri siano legali e corretti, esegue la richiesta, e ritorna il controllo all'istruzione che segue la system call.
- con l'istruzione di ritorno, il mode bit viene impostato a "user"

PROCESSOR AND O.S. MAIN FEATURES

Synchronous and asynchronous I/O

(a) Synchronous method

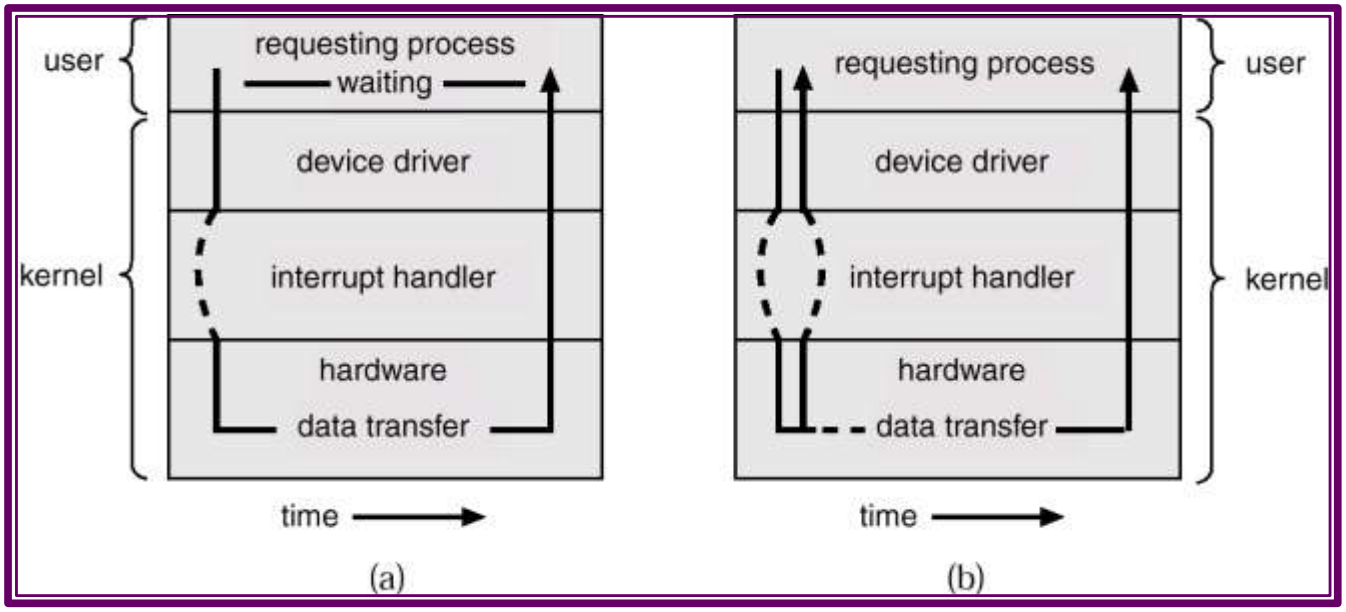
- After I/O starts, control returns to user program only upon I/O completion.
  - A special ***wait instruction*** idles the CPU until the next interrupt or a *tight wait loop* (***busy wait***) continues until an interrupt occurs, transferring control to another part of the O.S. (contention for memory access).
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing.

(b) Asynchronous method

- After I/O starts, control returns to user program without waiting for I/O completion.
  - A system call is requested to allow user to wait for I/O completion.
  - If no user or O.S. process needs for CPU, a wait instruction or wait loop is requested

PROCESSOR AND O.S. MAIN FEATURES

Synchronous and asynchronous I/O



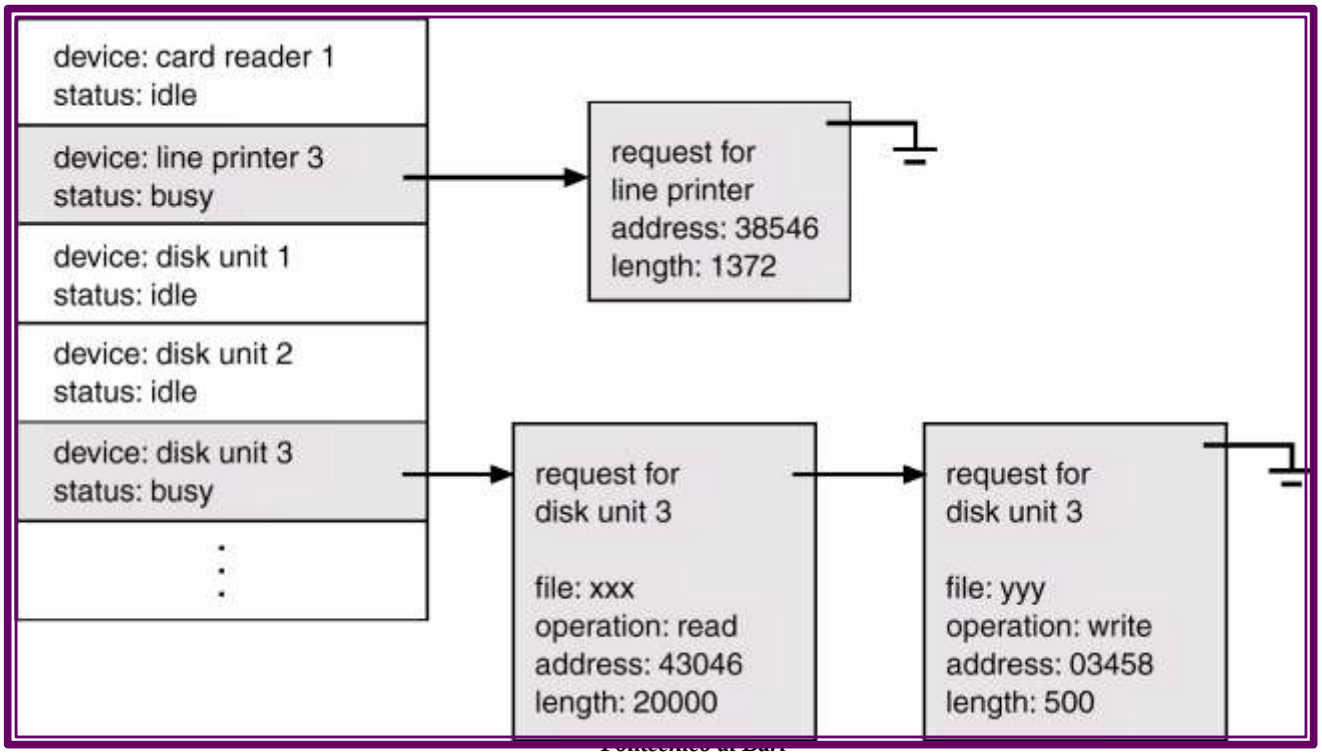
PROCESSOR AND O.S. MAIN FEATURES

Device Status Table

To override the problem of the wait instruction or of the tight loop, and to be able to keep track of many I/O requests at the same time, the O.S. uses a table, the Device Status table, containing an entry for each I/O device.

The O.S. indexes into I/O device table to determine device status and to modify table entry to include interrupt.

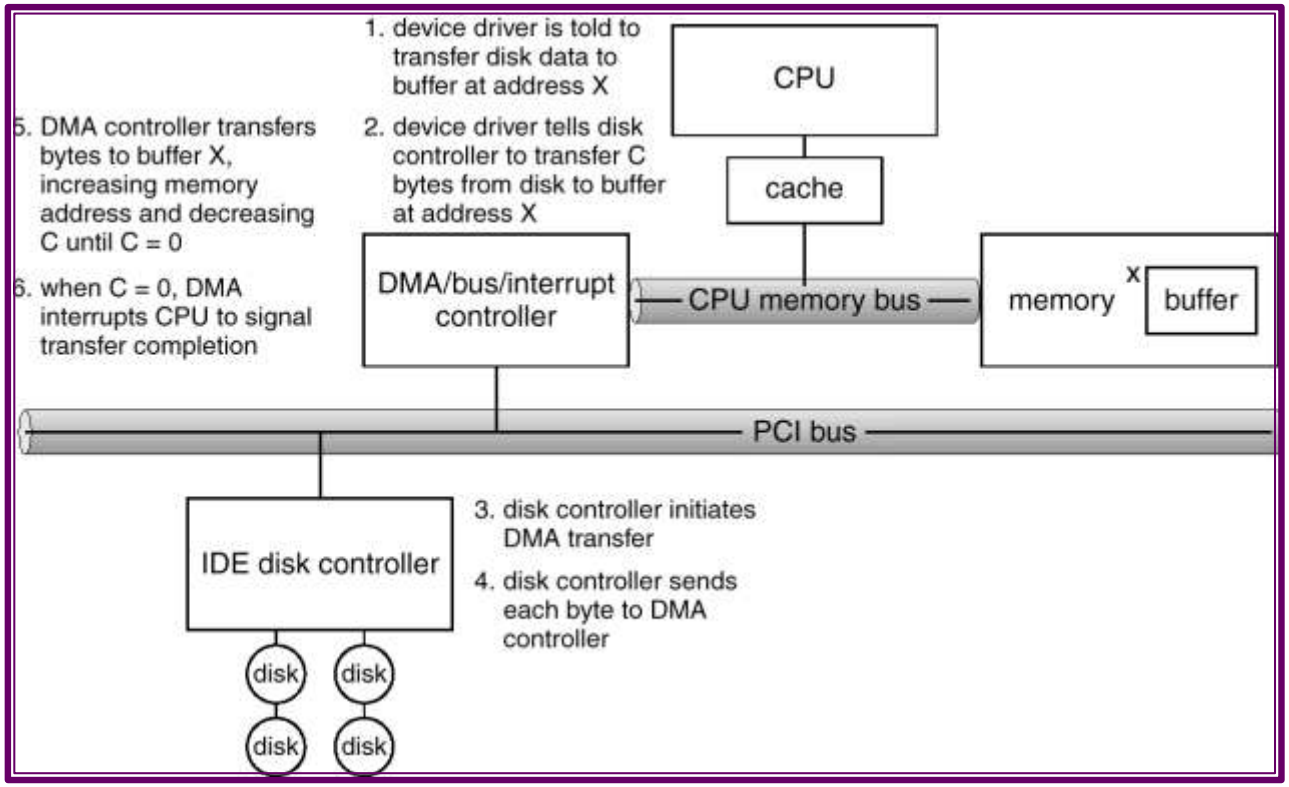
Device-status table contains entry for each I/O device indicating its type, address, and state.



PROCESSOR AND O.S. MAIN FEATURES

Direct Memory Access (DMA)

- Used for high-speed I/O devices, to avoid programmed I/O for large data movement.
- Able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than the one interrupt per byte.

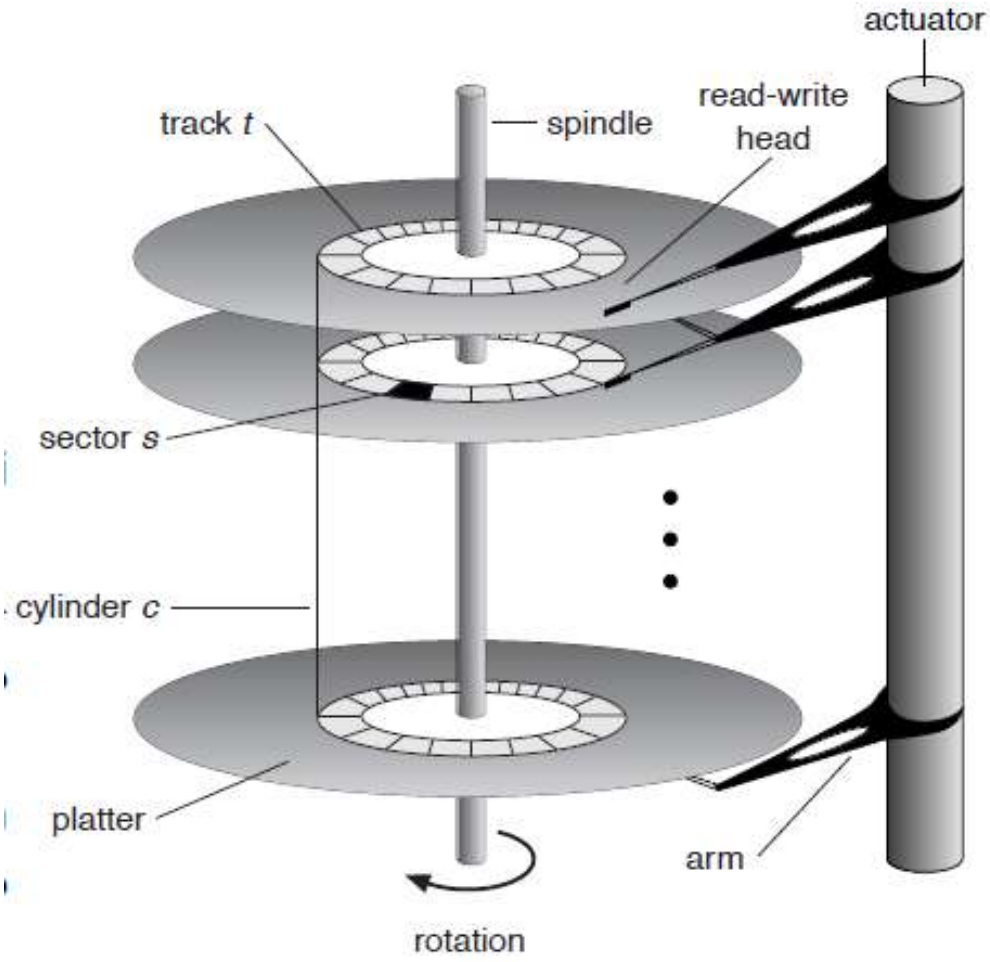


MEMORY TYPES

- ***Main memory*** – only large storage media that the CPU can access directly.
- ***Secondary storage*** – extension of main memory that provides large nonvolatile storage capacity.

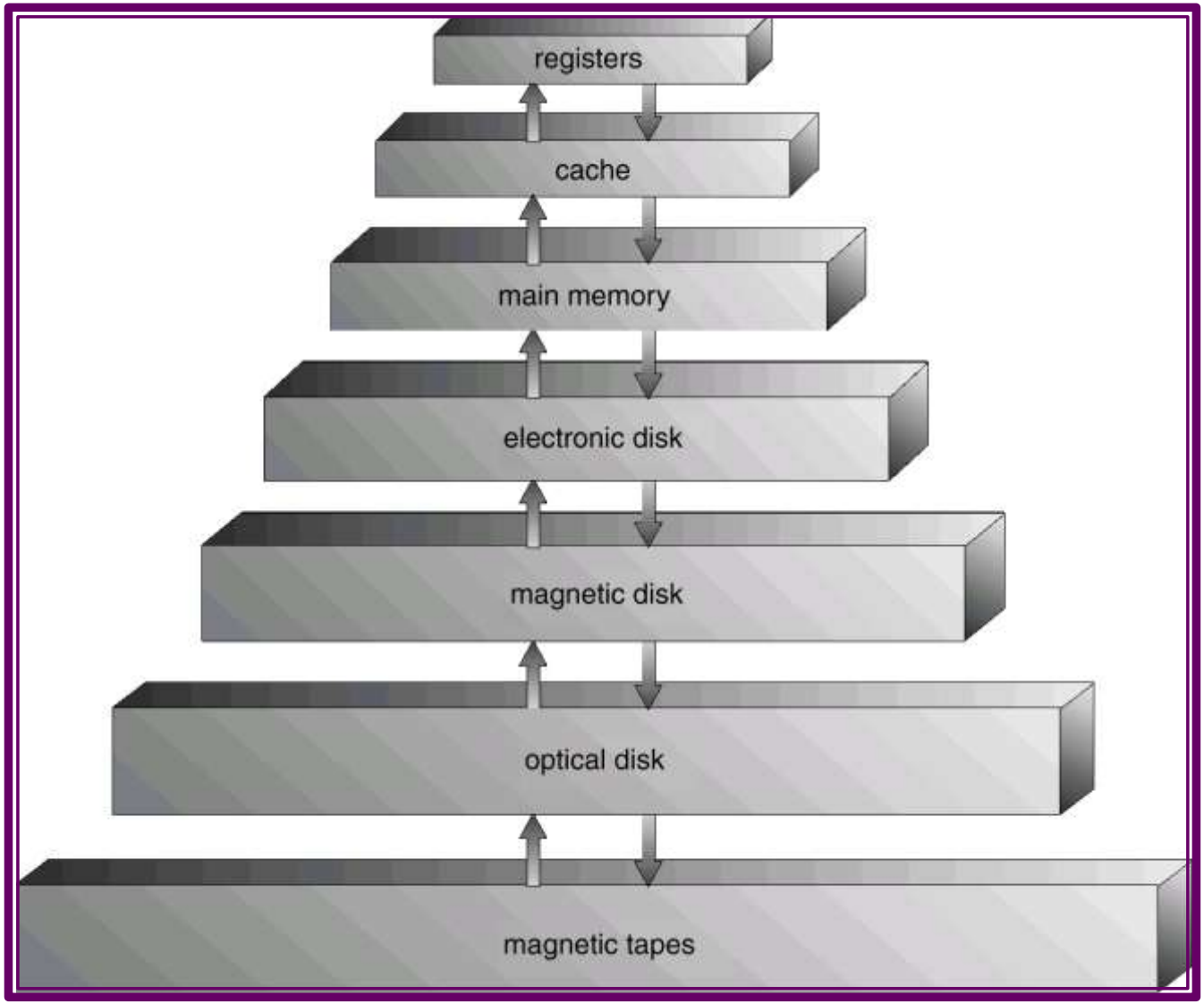
Most common secondary storage: **Magnetic disks** made from rigid *metal or glass platters* covered with iron-magnetic recording material

- Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
- The *disk controller* determines the logical interaction between the device and the computer.

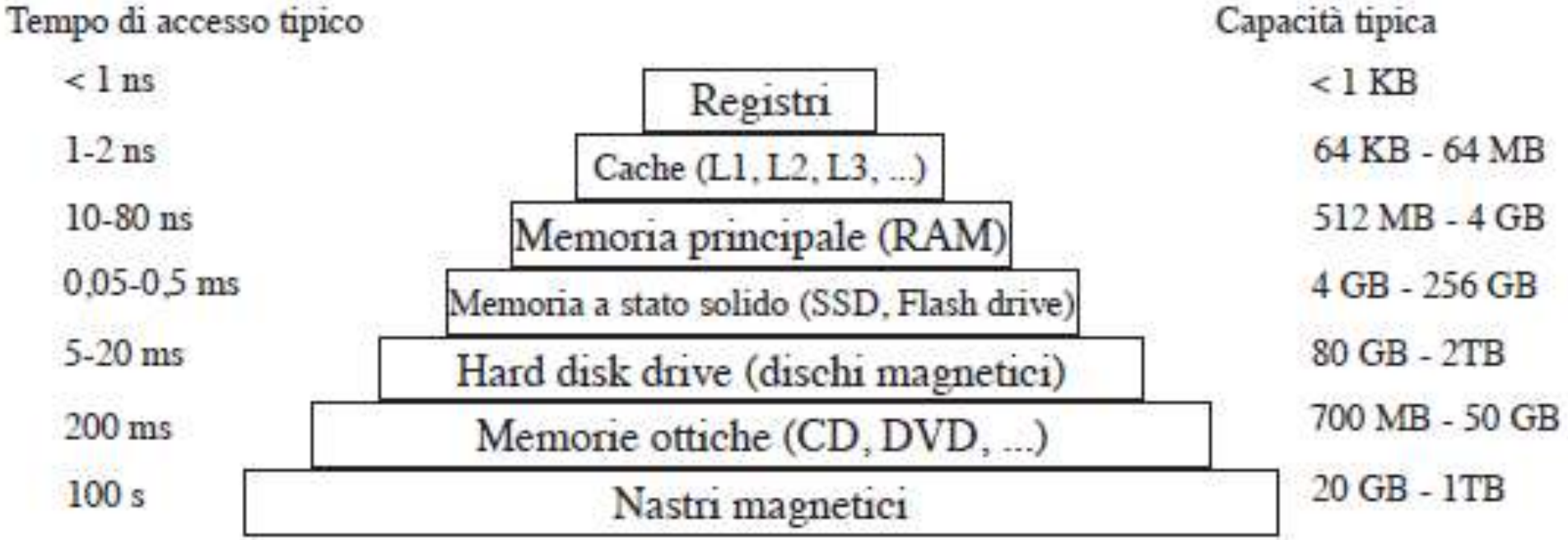


STORAGE DEVICE HIERARCHY

I sistemi di memorizzazione sono organizzati gerarchicamente, secondo *velocità*, *costo* e *volatilità*.

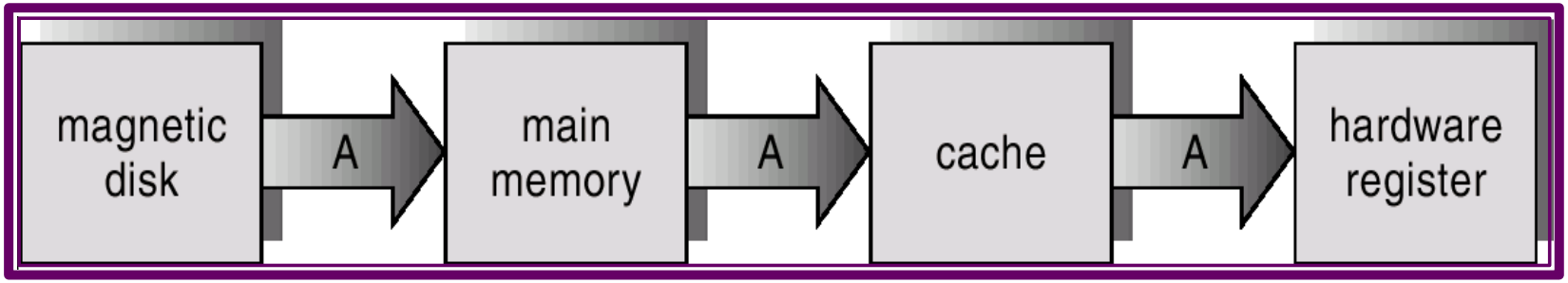


STORAGE DEVICE HIERARCHY



CACHE MEMORY

- Use of high-speed memory to hold recently-accessed data.
- Requires a *cache management* policy.
- Caching introduces another level in storage hierarchy. This requires data that is simultaneously stored in more than one level to be *consistent*.



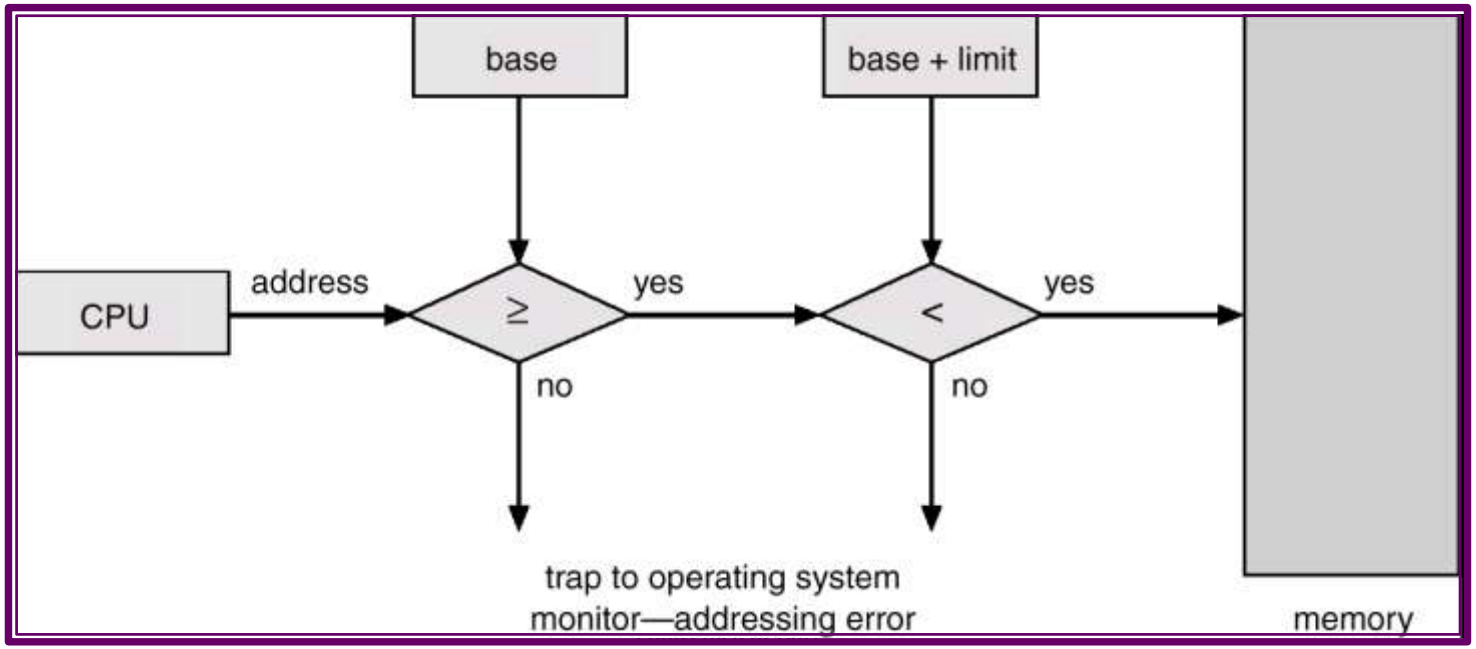
Una ***cache memory*** serve a duplicare i dati più frequentemente usati di una memoria, in una memoria più veloce.

La memoria principale può essere vista come una cache per la memoria secondaria.

HARDWARE PROTECTION

Memory Protection

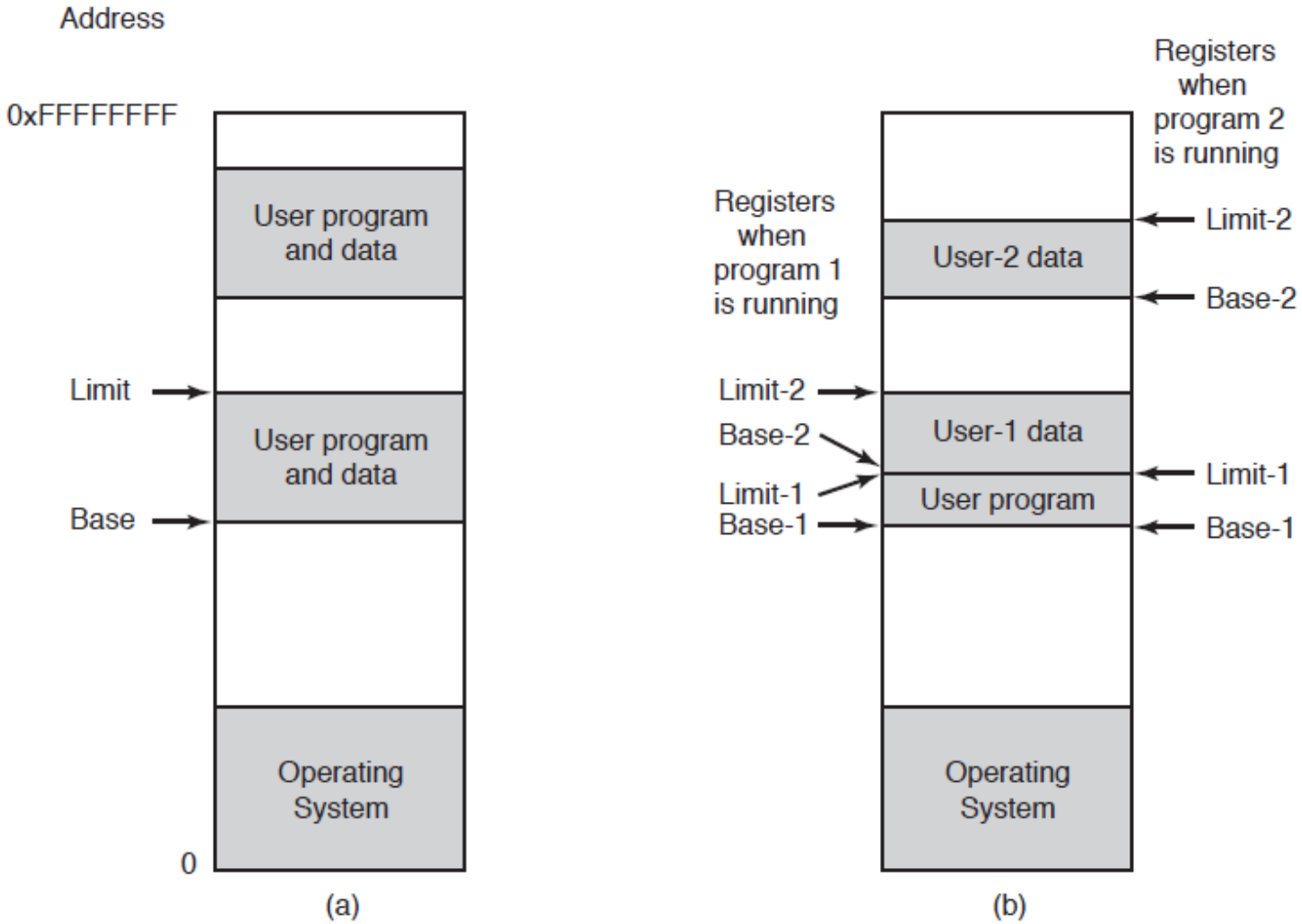
- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.



HARDWARE PROTECTION

Memory Protection

Even if 2 programs share the same program addresses but use 2 different data addresses

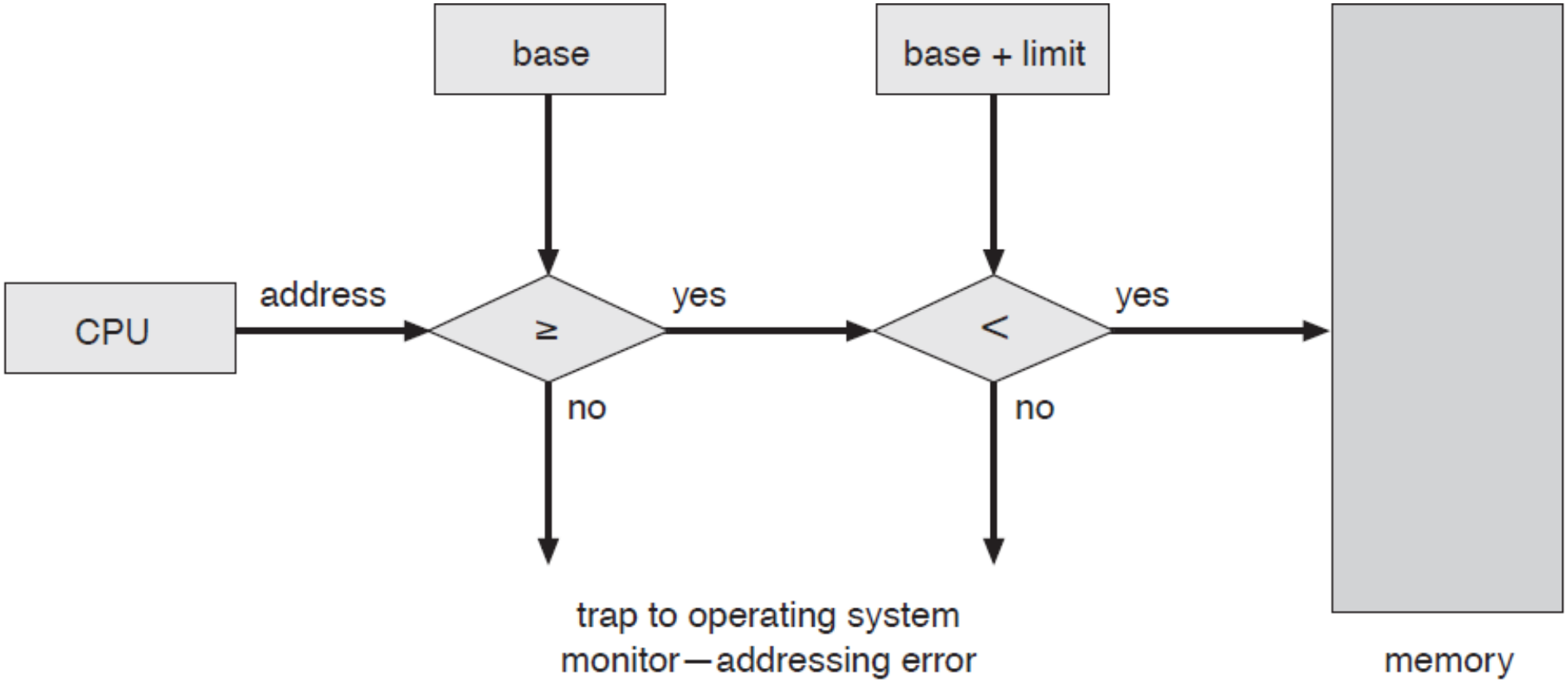


HARDWARE PROTECTION

Memory Protection

Essendo eseguito in modo monitor, il sistema operativo ha libero accesso a tutta la memoria, sia di sistema sia utente

Le istruzioni di caricamento dei registri base e limite sono privilegiate



HARDWARE PROTECTION

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every ***clock tick*** (1/50 of second, commonly).
  - When timer reaches the value 0, an interrupt occurs.
- Timer commonly used to implement time sharing.
- Time also used to compute the current time.
- Load-timer is a privileged instruction.