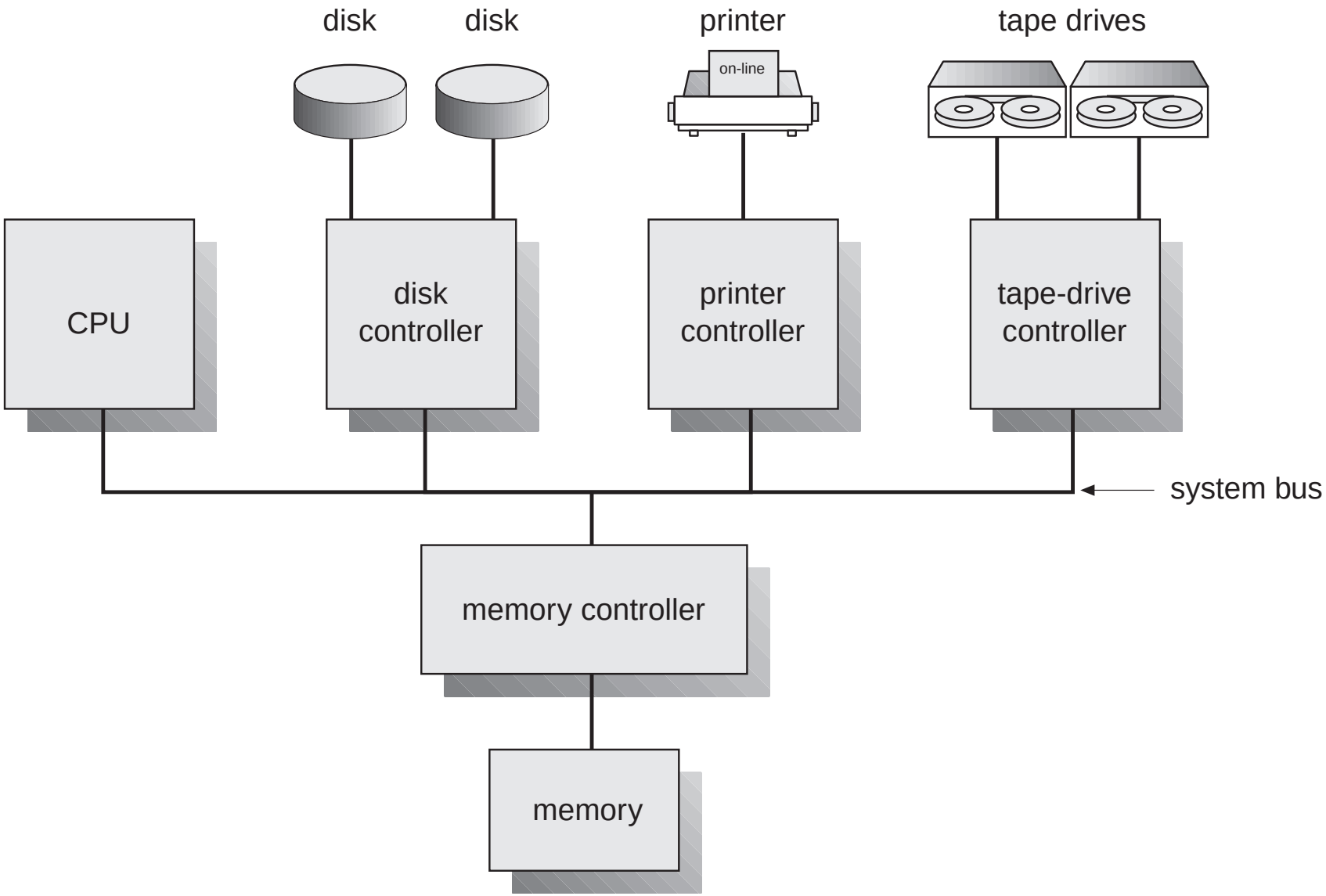
Struttura dei Sistemi di Calcolo

- Operazioni dei sistemi di calcolo
- Struttura dell'I/O
- Struttura della memoria
- Gerarchia delle memorie
- Protezione hardware
- Invocazione del Sistema Operativo

Architettura dei calcolatori



Operazioni dei sistemi di calcolo

- I dispositivi di I/O e la CPU possono funzionare concorrentemente
- Ogni controller 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 avviente tra il dispositivo e il buffer locale del controller
- Il controller informa la CPU quando ha terminato la sua operazione, generando un *interrupt*.

Funzionamento event-driven di un S.O.

- Il programma di *bootstrap* inizializza i registri della CPU e dei controller dei dispositivi di I/O

- Quindi carica in memoria il nucleo del sistema operativo e lo esegue

- Il sistema operativo lancia un processo speciale (init in Unix) e poi attende segnali di interrupt (eventi che possono cambiare il corso dell'esecuzione del programma corrente)

Interrupt

- Visono due tipi di interrupt
  - Segnali di interrupt che arrivano da un controller
  - Interrupt generate da software, chiamate *trap*: un trap può essere causato o da un errore o da una esplicita richiesta dell'utente, ad. es., una chiamata di una funzione di sistema (*system call*)

Funzioni comuni degli Interrupt

- Gli interrupt trasferiscono il controllo alla routine di servizio dell'interrupt, generalmente attraverso il *vettore di interruzioni*, che contiene gli indirizzi di tutte le routine di servizio.

- L'hardware deve salvare l'indirizzo dell'istruzione interrotta.

- Interrupt in arrivo sono *disabilitati* mentre un altro interrupt viene gestito, per evitare che vadano perduti.

Funzioni comuni degli Interrupt

- Gli interrupt trasferiscono il controllo alla routine di servizio dell'interrupt, generalmente attraverso il *vettore di interruzioni*, che contiene gli indirizzi di tutte le routine di servizio.

- L'hardware deve salvare l'indirizzo dell'istruzione interrotta.

- Interrupt in arrivo sono *disabilitati* mentre un altro interrupt viene gestito, per evitare che vadano perduti.

- Un *trap* è un interrupt generato da software, causato o da un errore o da una esplicita richiesta dell'utente.

- Un sistema operativo è *guidato da interrupt*

Gestione degli Interrupt

- Il sistema operativo preserva lo stato della CPU salvando registri e program counter.

- Determinazione di quale tipo di interrupt è avvenuto:

  – Se i controller non inviano direttamente segnali di interrupt alla CPU, si utilizza il *polling*: ciclicamente la CPU controlla lo stato (registri) dei dispositivi per vedere se si sono verificati eventi rilevanti (ad esempio fine di un trasferimento I/O) e quindi invoca la routine per gestire tale evento (ad es. trasferimento dei dati da buffer a memoria)

  – Se il controller invia un segnale di interrupt (con informazioni sul tipo di interrupt, ad es. un indice per il vettore delle interr.) il sistema operativo utilizza il selezionare la routine di gestione attraverso il vettore delle interruzioni

Struttura dell'I/O

- I/O sincrono: dopo che l'I/O è partito, il controllo ritorna al programma utente solo dopo che l'I/O è stato completato
  - l'istruzione hardware **wait** (se esiste) blocca la CPU fino alla prossima interruzione
  - oppure, la CPU aspetta la prossima intrruzione tramite un ciclo di attesa (chiamato *busy wait*)
  - in questo caso al più una richiesta di I/O è eseguita alla volta; non ci sono I/O paralleli

- I/O asincrono: dopo che l'I/O è partito, il controllo ritorna al programma utente senza aspettare che l'I/O venga completato

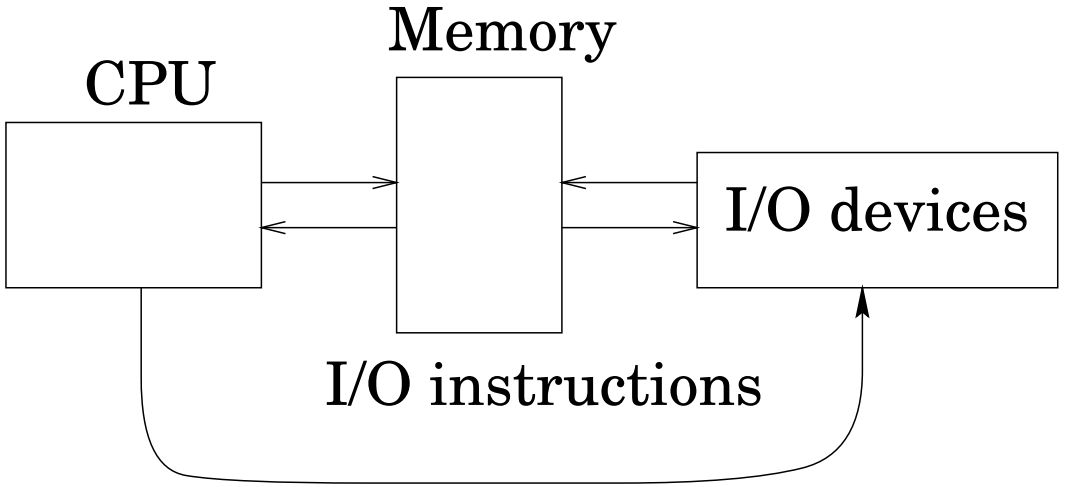
  – *chiamata di sistema* (*System call*) – richiede al sistema operativo di sospendere il processo in attesa del completamento dell'I/O.

  – Se non ci sono processi da eseguire la CPU esegue un'istruzione **wait**

  – una *tabella dei dispositivi* mantiene tipo, indirizzo e stato di ogni dispositivo di I/O.

  – Il sistema operativo accede alla tabella dei dispositivi per determinare lo stato e per mantenere le informazioni relative agli interrupt.

Struttura del Direct Memory Access (DMA)



- Usata per dispositivi in grado di trasferire dati a velocità prossime a quelle della memoria

- I controller trasferiscono blocchi di dati dal buffer locale direttamente alla memoria, senza intervento della CPU (per evitare di dover gestire troppe interruzioni)

- Viene generato un solo interrupt per blocco, invece di uno per ogni byte trasferito.

Struttura della Memoria

- Memoria principale (RAM) – la memoria che la CPU può accedere direttamente.

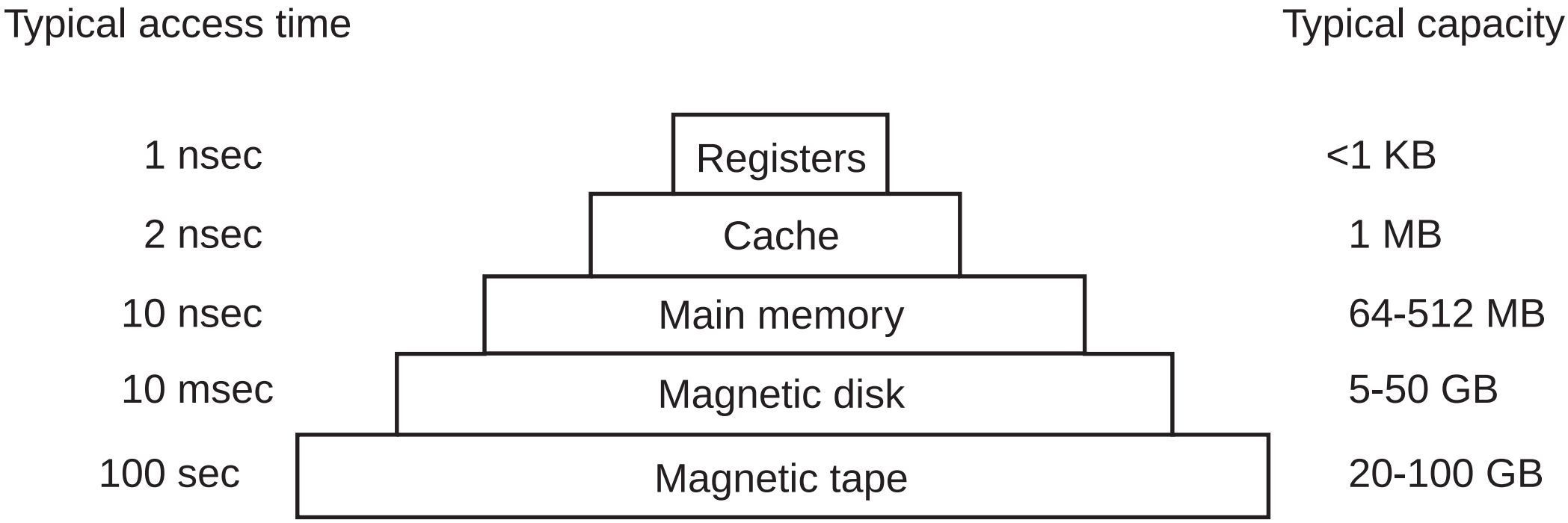
- Memoria secondaria (Dischi, floppy, CD, ...) – estensione della memoria principale che fornisce una memoria non volatile (e solitamente più grande)

Gerarchia della Memoria

I sistemi di memorizzazione sono organizzati gerarchicamente, secondo

- velocità
- costo
- volatilità

Caching – 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.


Typical access time
Typical capacity
1 nsec
Registers
<1 KB
2 nsec
Cache
1 MB
10 nsec
Main memory
64-512 MB
10 msec
Magnetic disk
5-50 GB
100 sec
Magnetic tape
20-100 GB

Protezione hardware

- Funzionamento in dual-mode
- Protezione dell'I/O
- Protezione della Memoria
- Protezione della CPU

Funzionamento Dual-Mode

- La condivisione di risorse di sistema richiede che il sistema operativo assicuri che un programma scorretto non possa portare altri programmi (corretti) a funzionare non correttamente.

- L'hardware deve fornire un supporto per differenziare almeno tra due modi di funzionamento

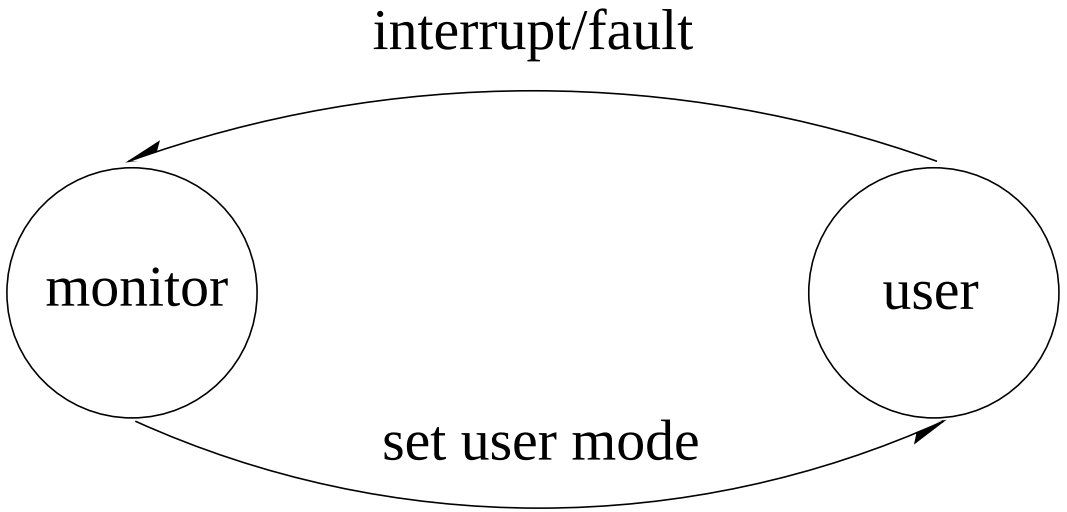
  1. *User mode* – la CPU sta eseguendo codice di un utente

  2. *Monitor mode* (anche *supervisor mode*, *system mode*, *kernel mode*) – la CPU sta eseguendo codice del sistema operativo

Funzionamento Dual-Mode (Cont.)

- La CPU ha un *Mode bit* che indica in quale modo si trova: supervisor (0) o user (1).

- Quando avviene un interrupt, l'hardware passa automaticamente in modo supervisore



- Le *istruzioni privilegiate* possono essere eseguite solamente in modo supervisore

Protezione dell'I/O

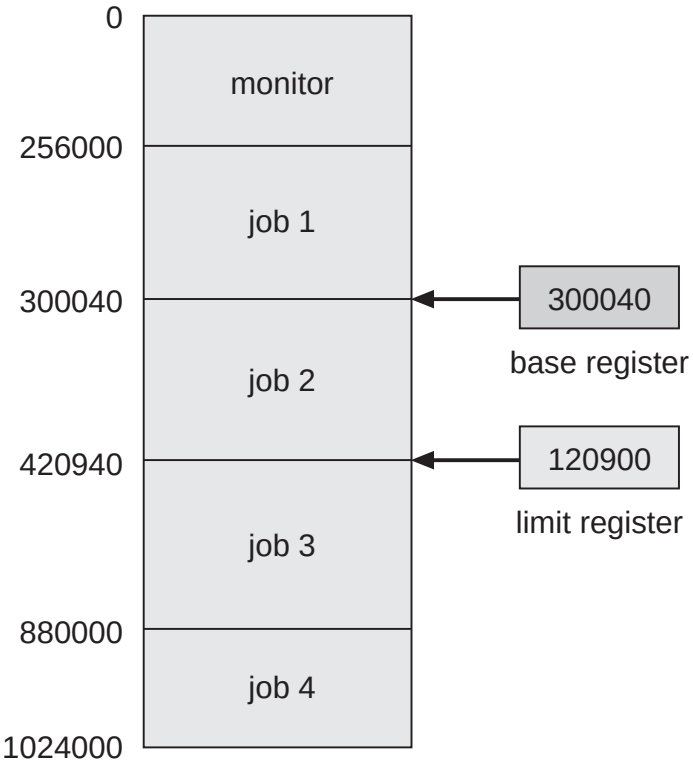
- Tutte le istruzione di I/O sono privilegiate

- Si deve assicurare che un programma utente non possa mai passare in modo supervisore (per esempio, andando a scrivere nel vettore delle interruzioni)

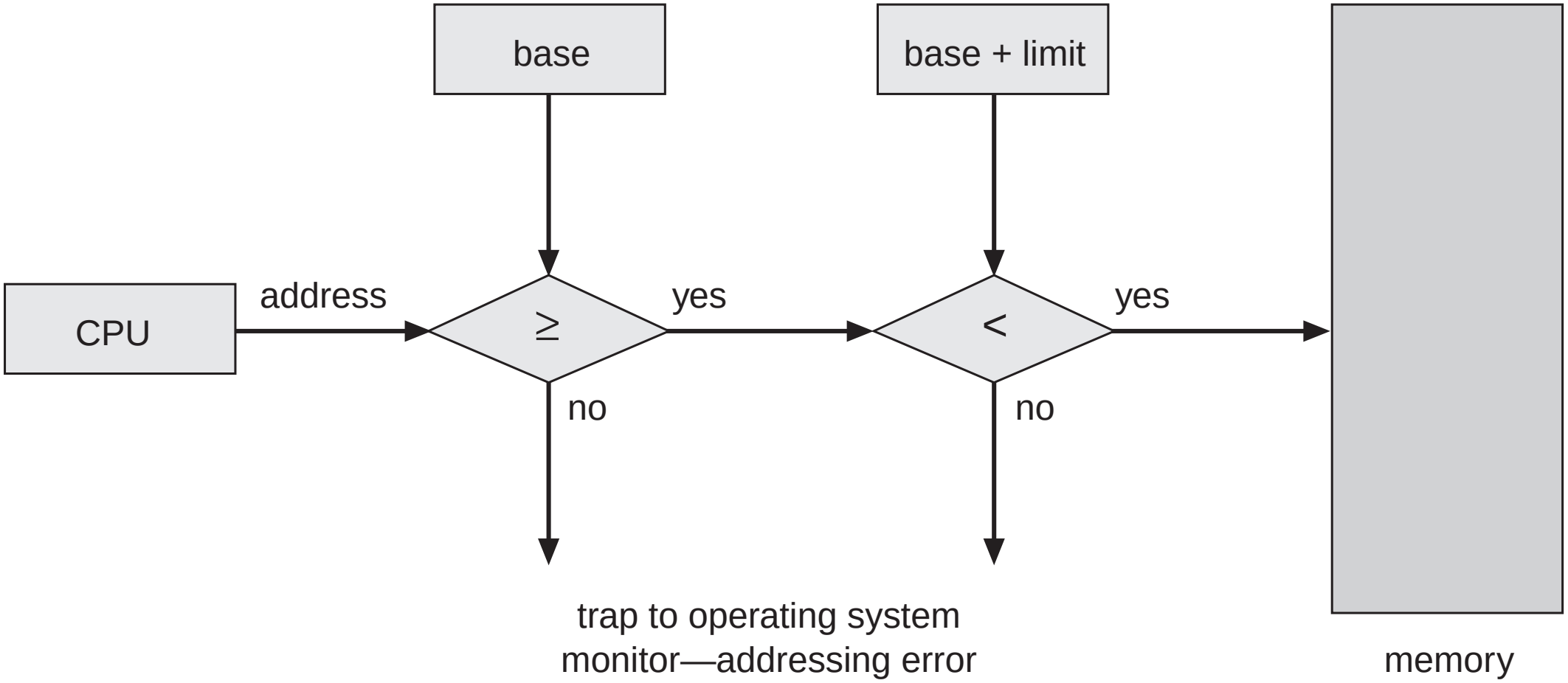
Protezione della Memoria

Si deve proteggere almeno il vettore delle interruzioni e le routine di gestione degli interrupt

- Per avere la protezione della memoria, si aggiungono due registri che determinano il range di indirizzi a cui un programma può accedere:
  registro base contiene il primo indirizzo fisico legale
  registro limite contiene la dimensione del range di memoria accessibile
- la memoria al di fuori di questo range è protetta



Protezione della Memoria (Cont.)



- 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

Protezione della CPU

- il *Timer* interrompe la computazione dopo periodi prefissati, per assicurare che periodicamente il sistema operativo riprenda il controllo
  - Il timer viene decrementato ad ogni *tick* del clock (1/50 di secondo, tipicamente)
  - Quanto il timer va a 0, avviene l'interrupt
- Il timer viene usato comunemente per implementare il time sharing
- Serve anche per mantenere la data e l'ora
- Il caricamento del timer è una istruzione privilegiata

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"