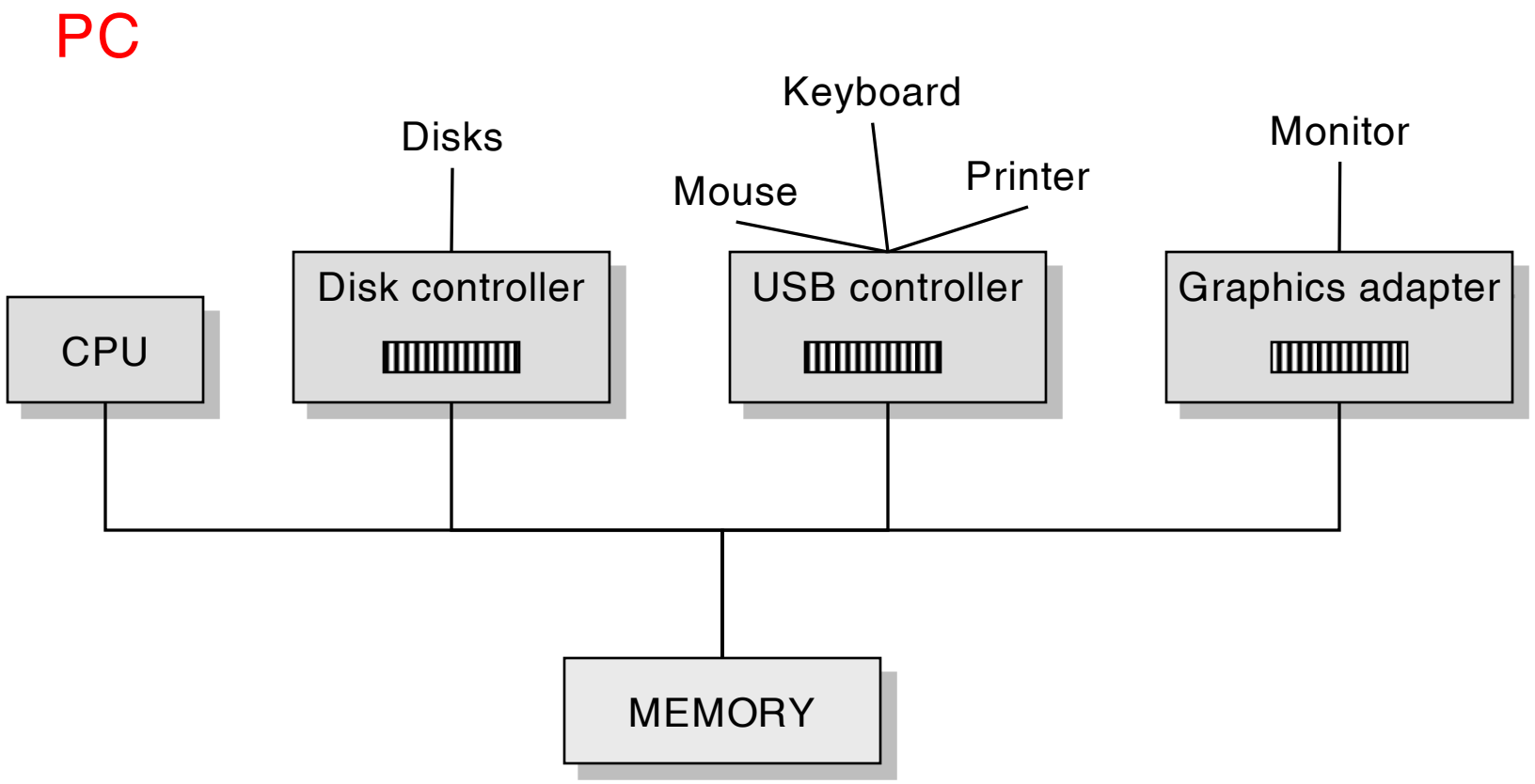
Chương 1 (bổ sung)

A. Hệ thống máy tính

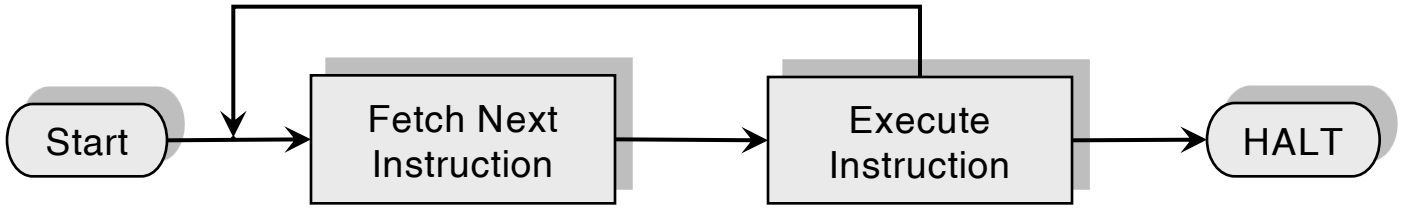
- Kiến trúc cơ bản của hệ thống máy tính
- Cơ chế vận hành của hệ thống
- Cấu trúc hệ thống xuất nhập (I/O)
- Cấu trúc và phân cấp hệ thống lưu trữ

Kiến trúc cơ bản của hệ thống máy tính

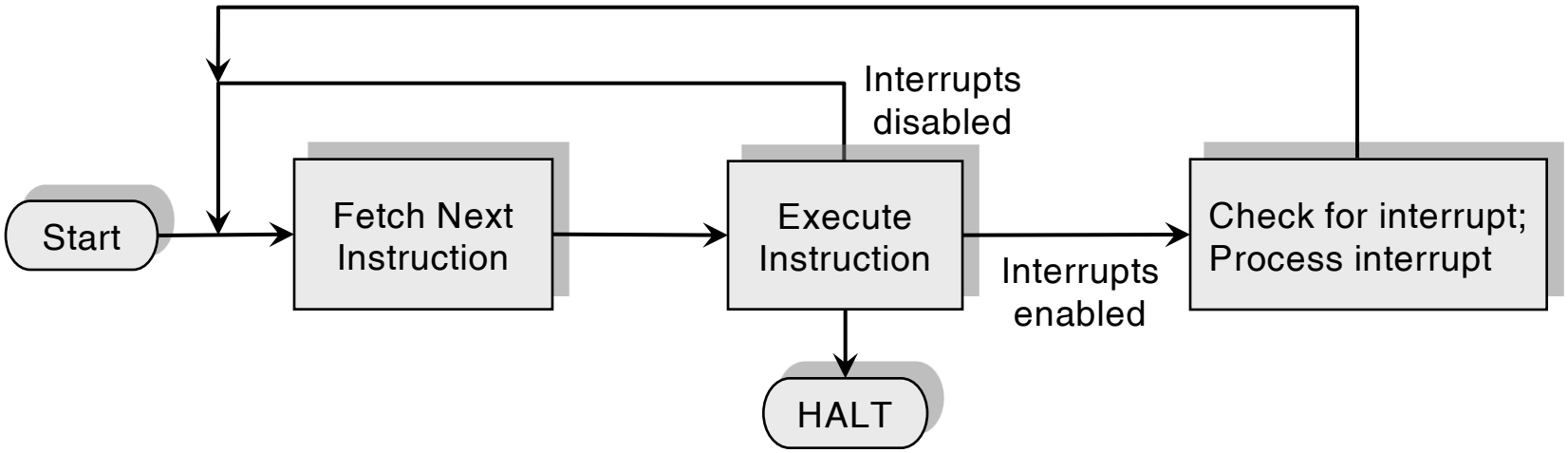


Đệm dữ liệu (local buffer)

Chu trình hoạt động của CPU



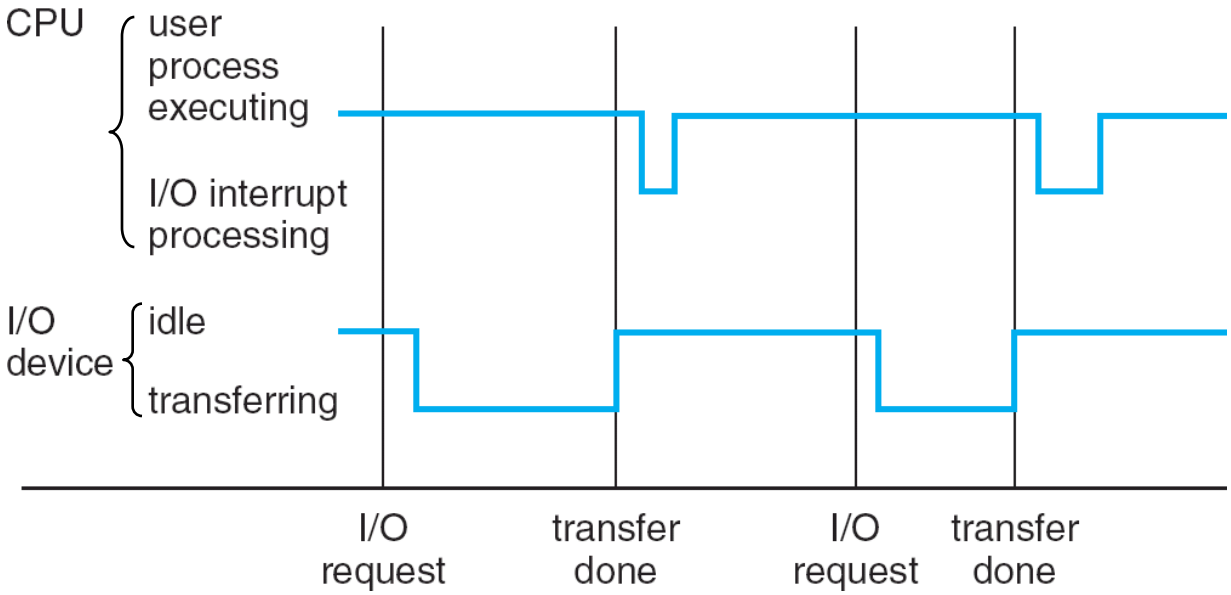
1. Chu trình đơn giản -- không có ngắt quãng



2. Chu trình có điều khiển ngắt quãng

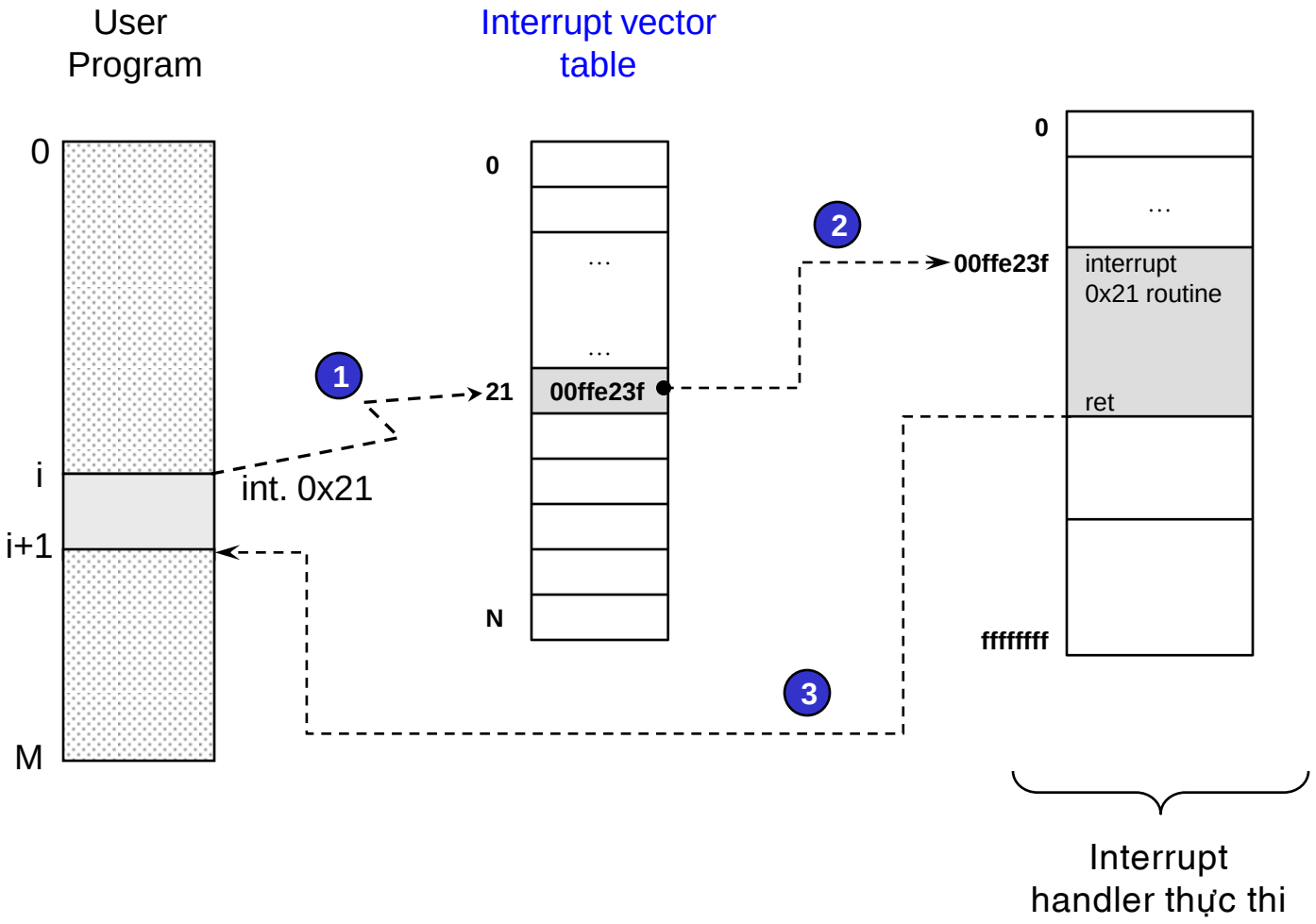
Ngắt quãng

- Phân loại: ngắt quãng do
  - *Program:* tràn số học, chia cho 0, truy cập bộ nhớ bất hợp pháp
  - *Timer:* cho phép CPU thực thi một tác vụ nào đó theo định kỳ
  - *I/O:* kết thúc tác vụ I/O, xảy ra lỗi trong I/O
  - *Hardware failure:* Hư hỏng nguồn, lỗi memory parity,…
  - *Trap (software interrupt):* yêu cầu dịch vụ hệ thống (gọi system call),…

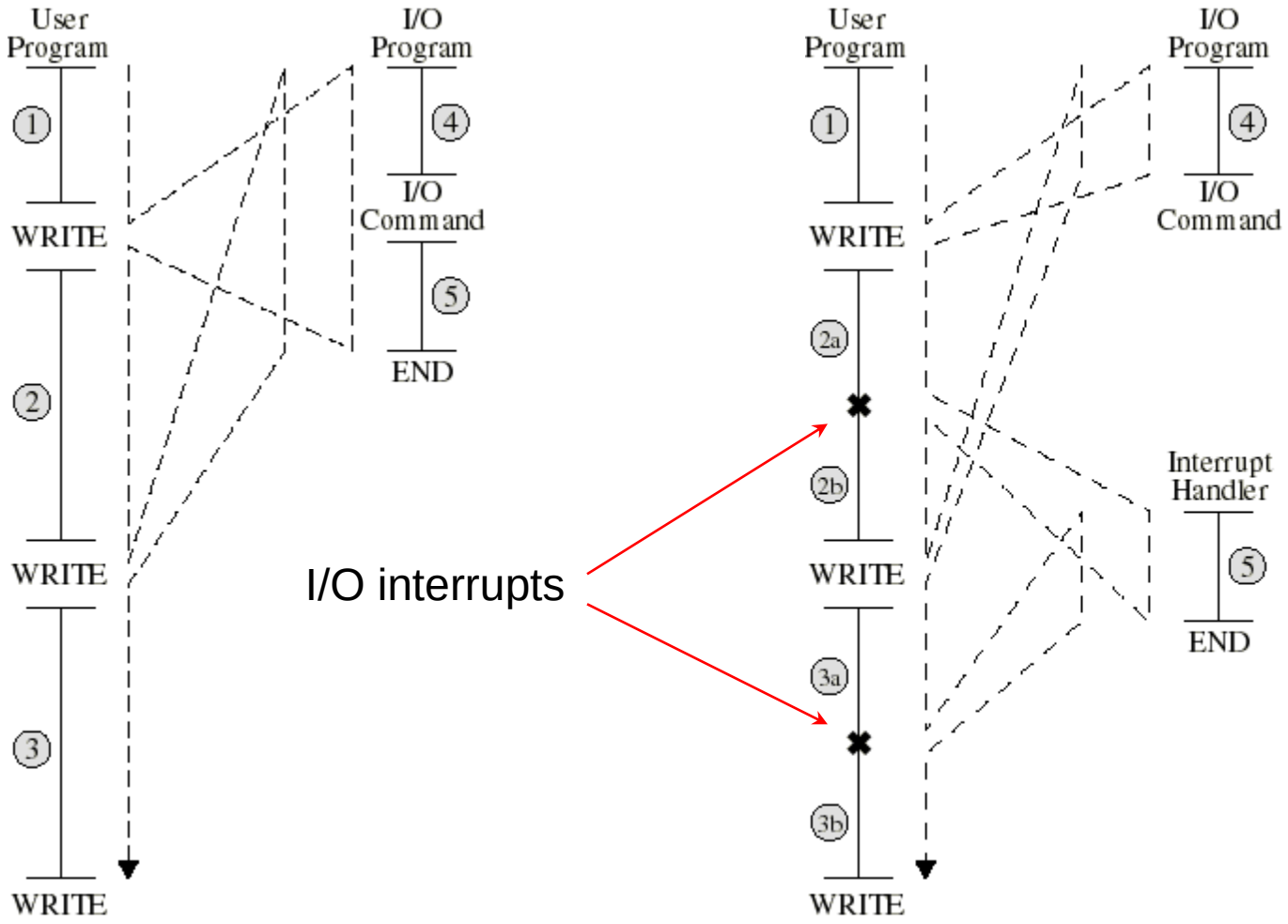


Lược đồ thời gian khi process có yêu cầu các tác vụ I/O

Quá trình xử lý ngắt quãng



Quá trình xử lý ngắt quãng (tt)



Không sử dụng ngắt quãng

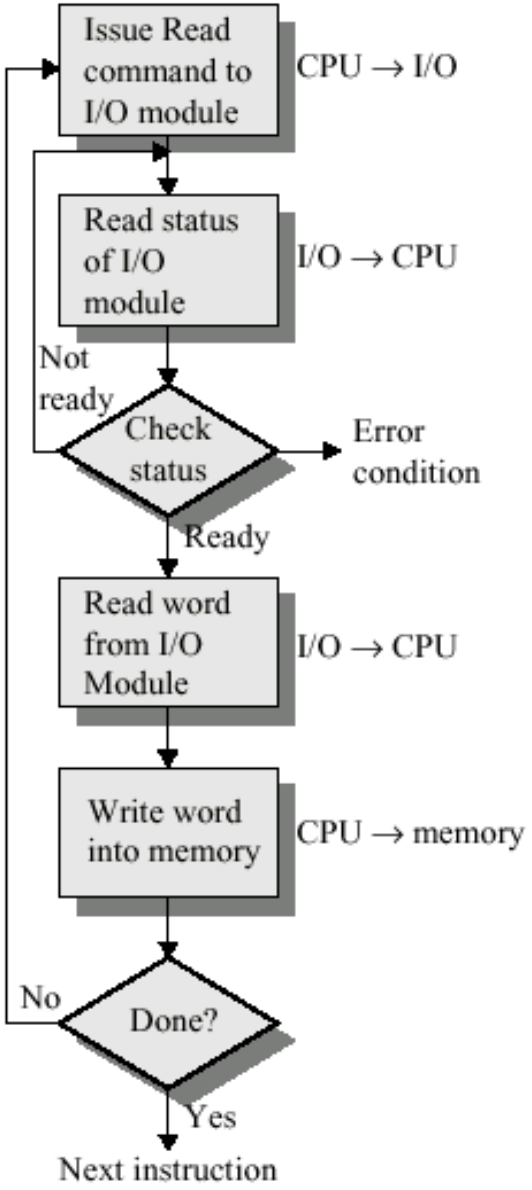
Sử dụng ngắt quãng

(hình chỉ minh họa I/O interrupt)

Cấu trúc hệ thống I/O

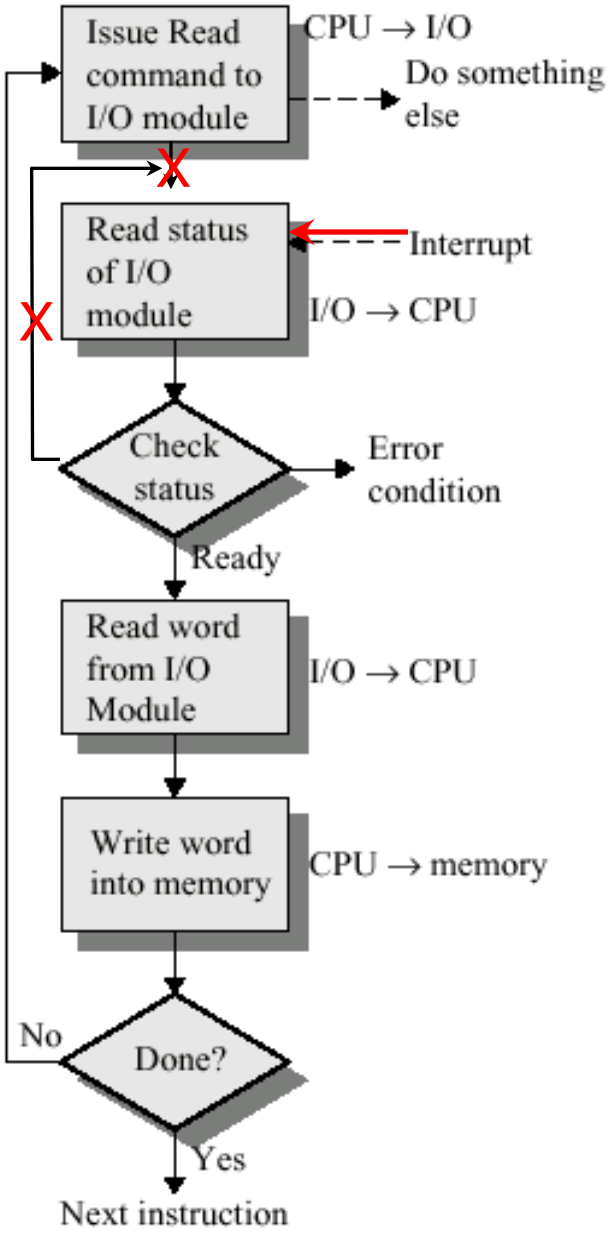
Các kỹ thuật thực hiện I/O

- *Polling*, ví dụ CPU đọc dữ liệu:
  - Để đọc dữ liệu từ một thiết bị I/O (thông qua I/O port), CPU thiết lập một bit (bit ← 1) của thanh ghi điều khiển (control register) để báo hiệu lệnh đọc cho I/O controller.
  - I/O controller đọc word dữ liệu từ thiết bị I/O, xóa bit điều khiển (bit ← 0)
  - I/O controller không gây ra ngắt mỗi khi xong việc. CPU phải đọc status bit (*polling*) để kiểm tra trạng thái thiết bị I/O
  - Khi I/O controller sẵn sàng, CPU đọc word dữ liệu từ thanh ghi dữ liệu (data register); CPU gửi lệnh đọc word kế.



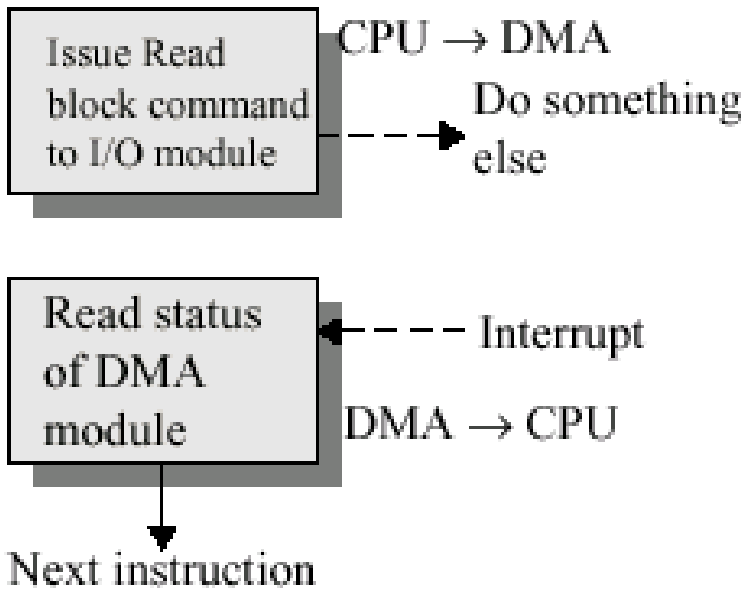
Các kỹ thuật thực hiện I/O (tt)

- *Interrupt-driven I/O*
  - CPU không poll mà I/O controller sẽ gây ra ngắt quãng mỗi khi xong và sẵn sàng cho tác vụ I/O mới.
  - Trong lúc thiết bị I/O thực thi lệnh, CPU có thể thực thi công việc khác.
  - Polling và interrupt-driven I/O đều tiêu tốn thời gian xử lý của CPU bởi vì CPU phải copy byte dữ liệu được đọc/ghi ↔ memory (*programmed I/O*, PIO).
  - Thích hợp cho các thiết bị I/O có tốc độ không cao (keyboard, mouse)



Các kỹ thuật thực hiện I/O (tt)

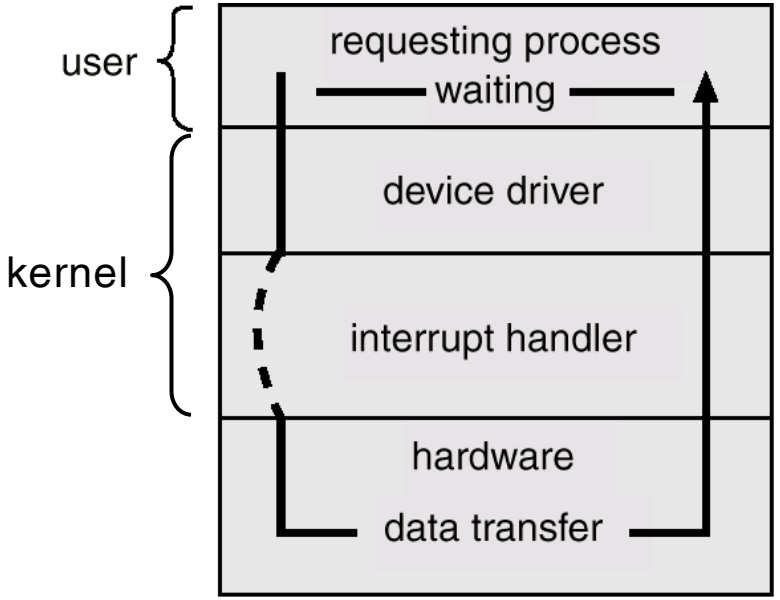
- *Direct Memory Access* (DMA)
  - CPU gửi yêu cầu đến module DMA (= DMA controller)
  - Module DMA chuyển một khối dữ liệu giữa bộ nhớ và thiết bị I/O mà không cần CPU can thiệp.
  - Khi xong một tác vụ gửi / nhận thì phát khởi một ngắt quãng.
  - CPU chỉ tham gia vào giai đoạn khởi đầu và kết thúc của việc truyền / nhận dữ liệu
  - Trong khi đang truyền / nhận dữ liệu, CPU có thể thực thi công việc khác
  - Thích hợp cho các thiết bị có tốc độ cao (đĩa)



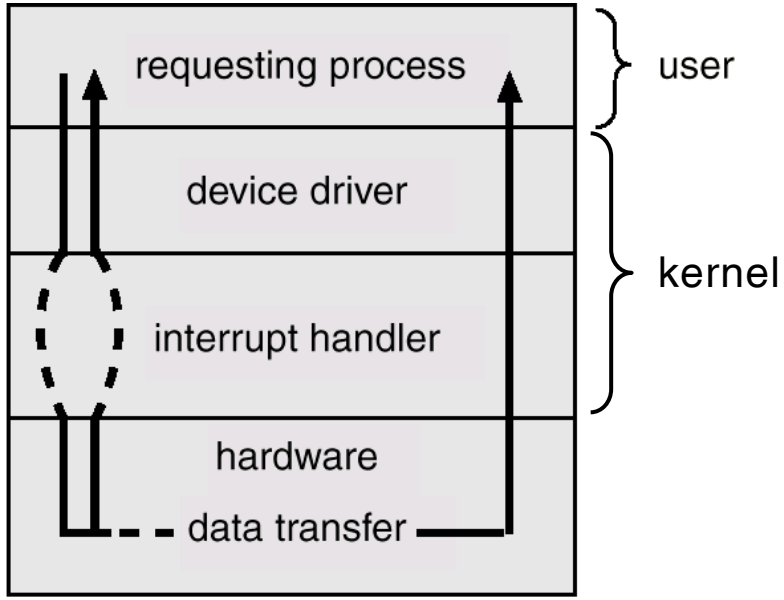
Các kỹ thuật thực hiện I/O (tt)

- Phương pháp thực hiện I/O

Synchronous



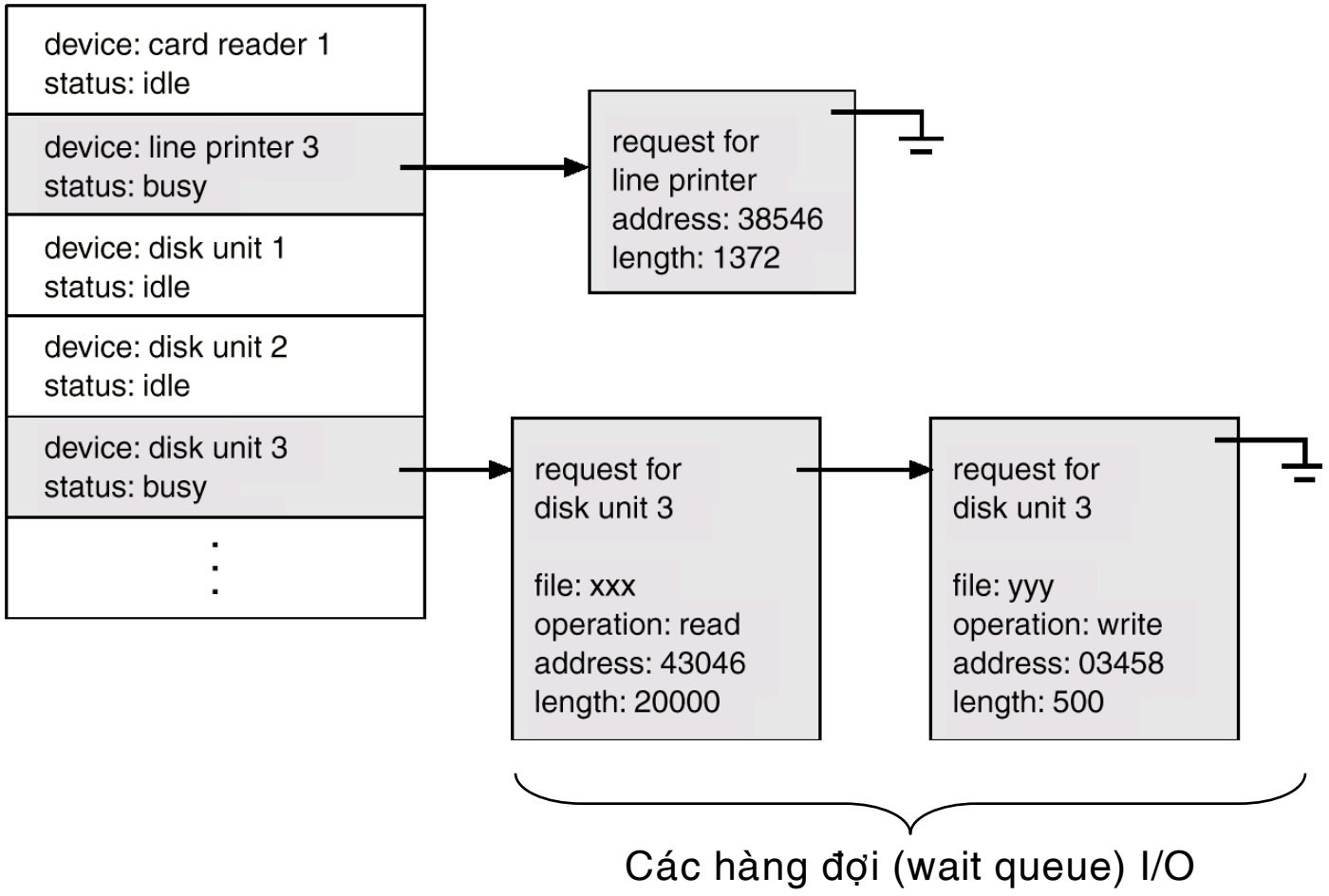
Asynchronous



- - - : “bypassing”

Các kỹ thuật thực hiện I/O (tt)

- Hàng đợi các yêu cầu I/O, vd

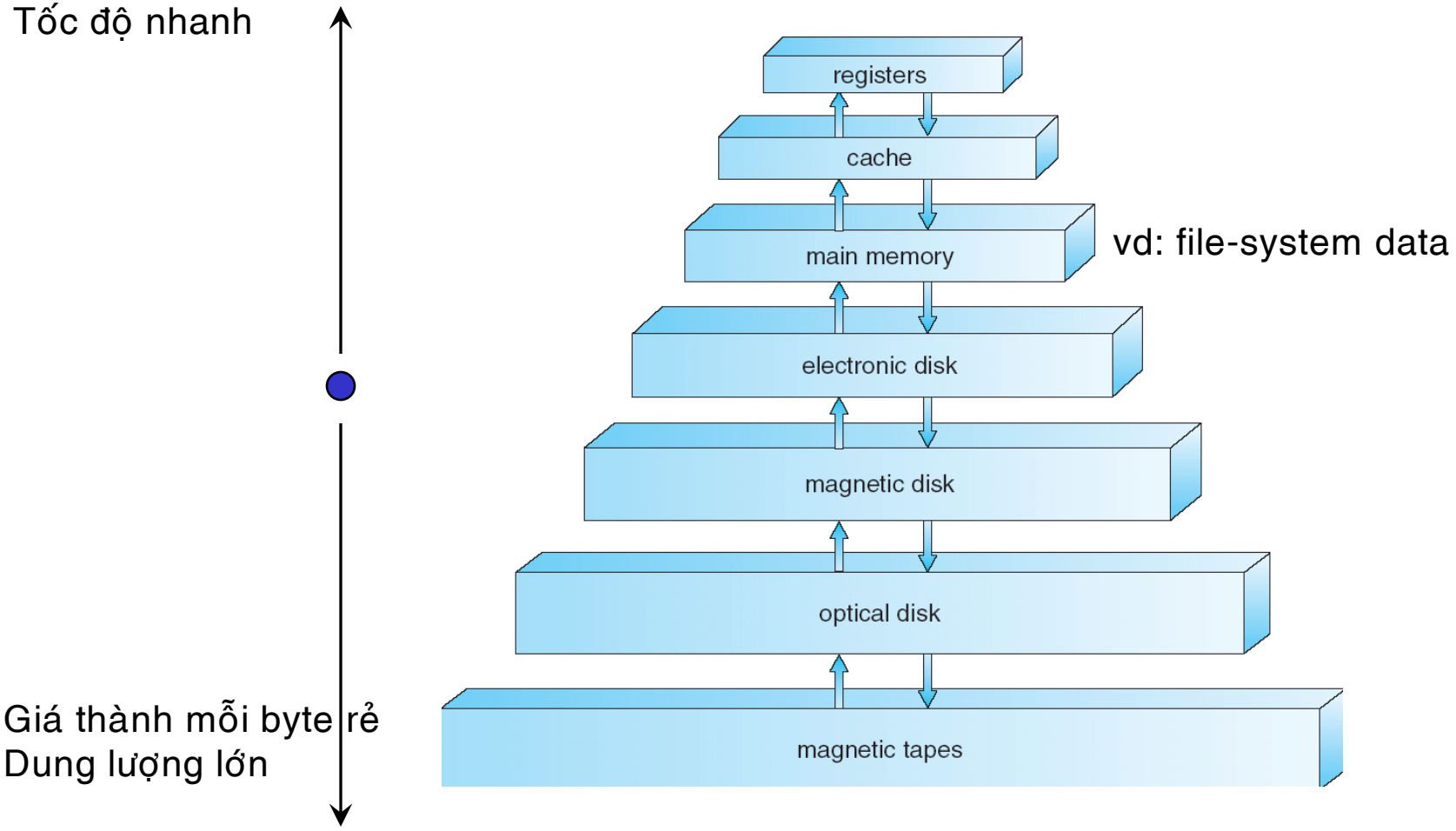


Cấu trúc & phân cấp hệ thống lưu trữ

Hệ thống lưu trữ

- Lưu trữ (memory, storage) là một trong những dạng thức I/O quan trọng
  - *Bộ nhớ chính* (main memory, primary memory)
    - CPU chỉ có thể truy cập trực tiếp thanh ghi (registers) và bộ nhớ ROM, RAM
  - *Bộ nhớ phụ* (secondary storage)
    - Hệ thống lưu trữ thông tin *bền vững* (nonvolatile storage)
    - Đĩa từ (magnetic disk): đĩa mềm, đĩa cứng, băng từ
    - Đĩa quang (optical disk): CD-ROM, DVD-ROM
    - Flash ROM: USB disk

Phân cấp hệ thống lưu trữ

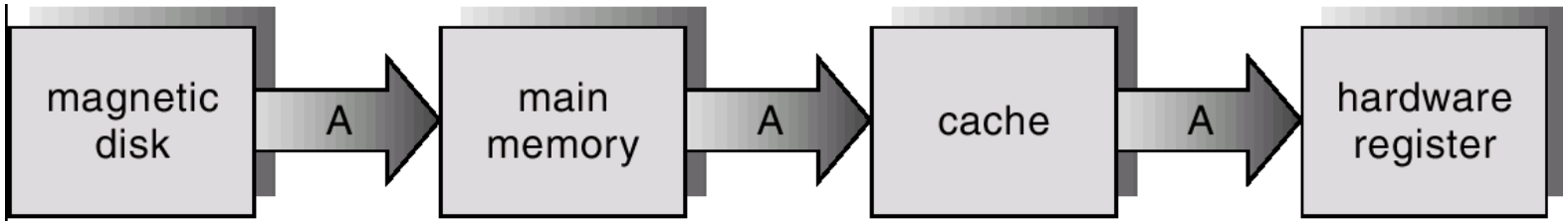


Phân cấp hệ thống lưu trữ

- Mục tiêu
  - Giá thành mỗi byte thấp gần với mức lưu trữ rẻ nhất
  - Tốc độ nhanh gần với mức lưu trữ nhanh nhất

Kỹ thuật caching

- *Caching*
  - nạp trước dữ liệu vào thiết bị lưu trữ tốc độ cao hơn
- Tại sao dùng cache?
  - Chênh lệch lớn giữa tốc độ CPU và tốc độ bộ nhớ RAM, đĩa,…
- Vì sao caching "works"? → nguyên lý cục bộ (locality principle)
- Dữ liệu lớn, còn kích thước cache nhỏ → phải quản lý cache: thay nội dung cache
- Trong kỹ thuật caching, một dữ liệu có thể được lưu trữ nhiều nơi → cần bảo đảm tính nhất quán dữ liệu: *cache coherency problem*



A: dữ liệu

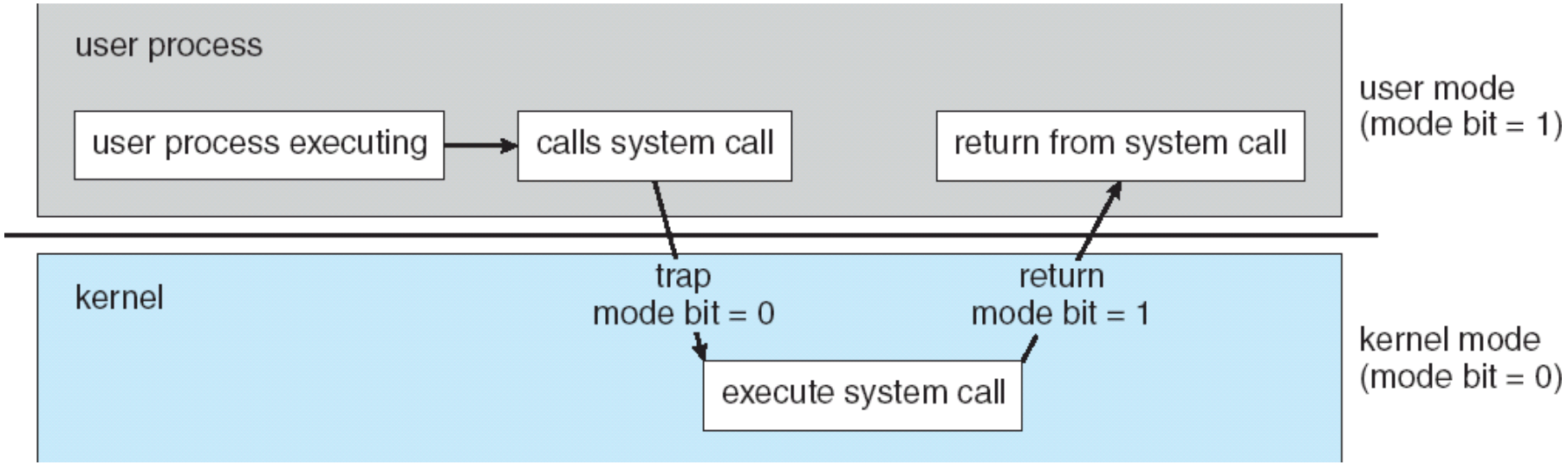
Dual mode

- Mục đích: bảo vệ hệ điều hành và chương trình ứng dụng
- Giải pháp

  Kỹ thuật *dual mode:* cần có phần cứng hỗ trợ
  - *User mode* – thực thi với quyền hạn của user bình thường
  - *Kernel mode* (còn gọi là supervisor mode, system mode, monitor mode) – có toàn quyền truy xuất tài nguyên hệ thống

Dual mode (tt)

- Phần cứng có thêm *mode bit* để kiểm soát mode hiện hành:
  - mode bit = 0: kernel mode
  - mode bit = 1: user mode
  - Đang ở user mode, nếu CPU bị ngắt (do thiết bị ngoại vi, do lỗi xảy ra,…), CPU sẽ chuyển sang kernel mode và thực thi interrupt service routine tương ứng.



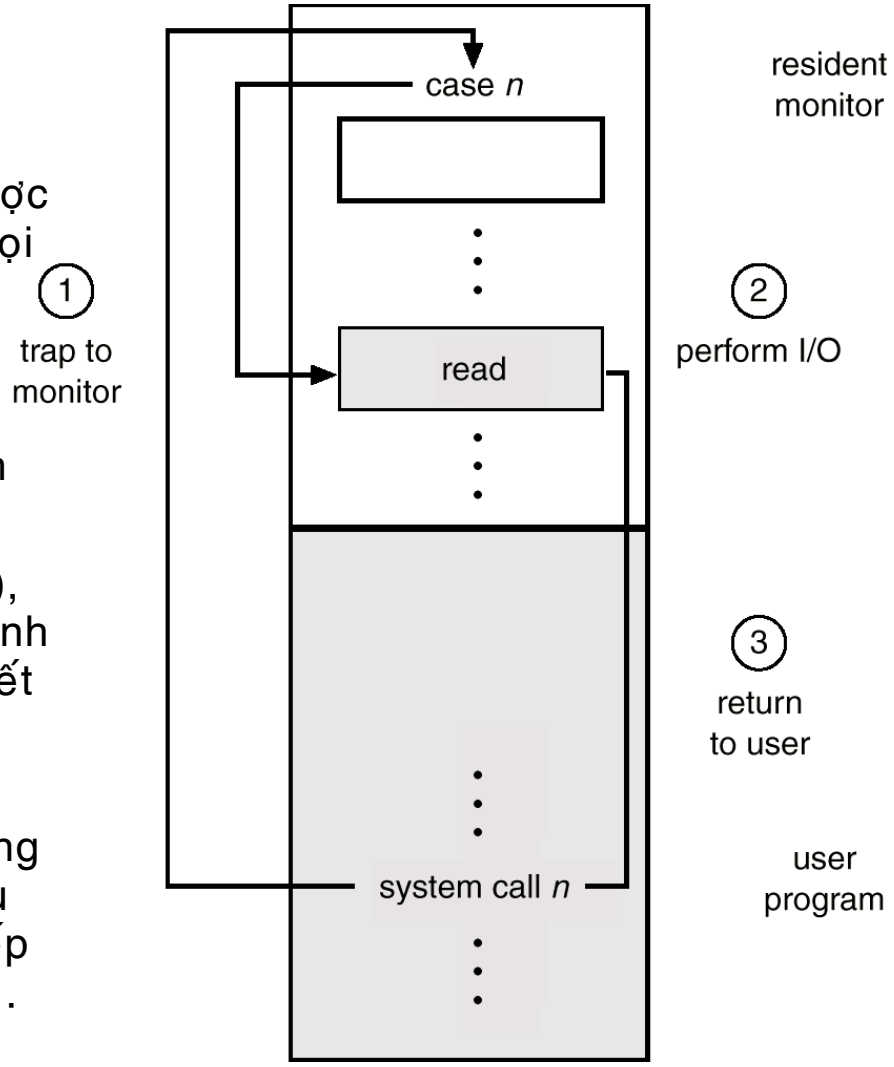
Dòng thực thi và thay đổi mode khi gọi system call

Bảo vệ phần cứng – Lệnh I/O

- Giải pháp: lệnh I/O đều là *privileged instruction*
  - User mode program không thực thi được lệnh I/O (→ trap), phải thông qua lời gọi *system call*

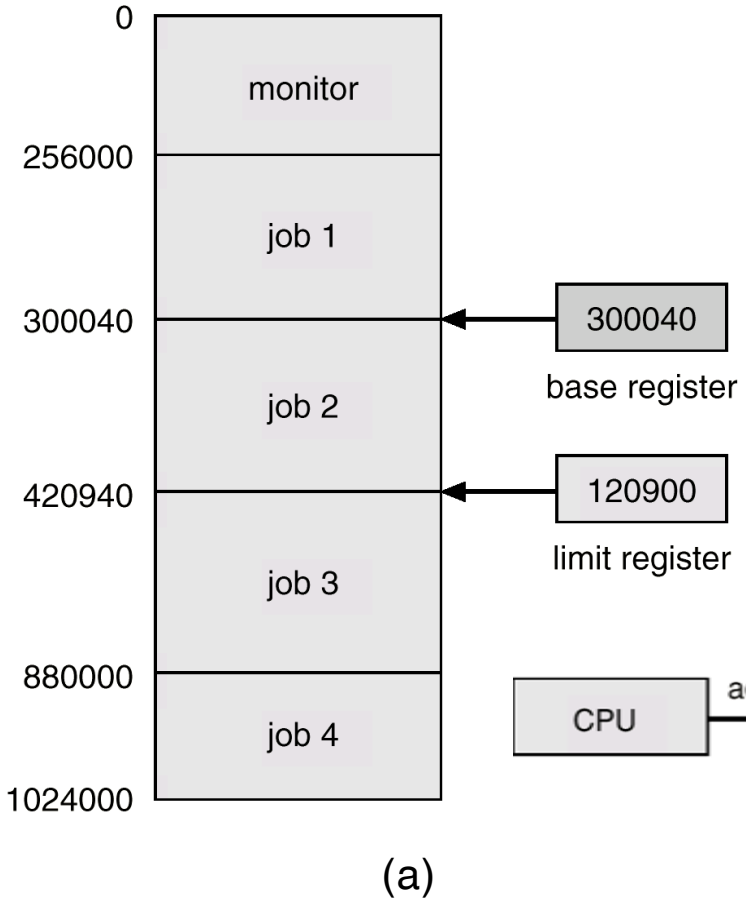
System call

- Là phương thức duy nhất để process yêu cầu các dịch vụ của hệ điều hành
- System call sẽ gây ra ngắt mềm (*trap*), quyền điều khiển được chuyển đến trình phục vụ ngắt tương ứng, đồng thời thiết lập *mode* = 0 (kernel mode).
- Hệ điều hành kiểm tra tính hợp lệ, đúng đắn của các đối số, thực hiện yêu cầu rồi trả quyền điều khiển về lệnh kế tiếp ngay sau lời gọi system call, *mode* = 1.



Bảo vệ phần cứng – Bộ nhớ

Vd: bảo vệ bộ nhớ dùng 2 thanh ghi

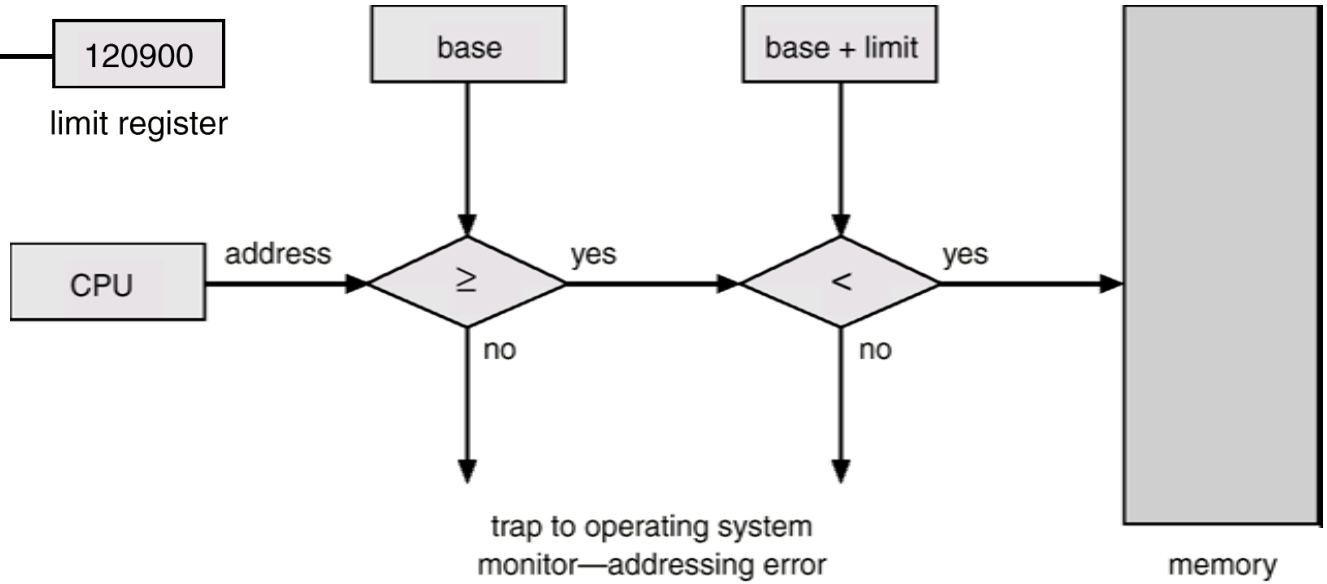


(a)

- Truy cập bộ nhớ ngoài vùng xác định bởi thanh ghi base và thanh ghi limit sẽ sinh ra trap

- Lệnh nạp giá trị cho các thanh ghi base và thanh ghi limit đều là privileged instruction

(b)



Bảo vệ phần cứng – CPU

- Bảo vệ CPU
  - Bảo đảm OS duy trì được quyền điều khiển
  - Tránh trường hợp CPU bị kẹt trong các vòng lặp vô hạn

  Cơ chế thực hiện là dùng timer để kích khởi các ngắt quãng định kỳ
  - Bộ đếm timer sẽ giảm dần sau mỗi xung clock.
  - Khi bộ đếm timer bằng 0 thì ngắt timer được kích hoạt → hệ điều hành sẽ nắm quyền điều khiển.

- Lệnh nạp giá trị bộ đếm timer là một privileged instruction.

Timer

- Có thể sử dụng timer để thực hiện time-sharing.
  - Thiết lập timer gây ngắt định kỳ *N* ms (*N: time slice*, *quantum time*) và định thời CPU sau mỗi lần ngắt.

- Có thể dùng timer để tính thời gian trôi qua (elapse time)