

NGUYÊN LÝ HỆ ĐIỀU HÀNH

Giảng viên: TS. Đoàn Thị Quế
Bộ môn Mạng và an toàn thông tin

Chương 5: Quản lý vào ra

- Giới thiệu về quản lý vào/ra
- Nguyên lý của phần cứng vào/ra
- Nguyên lý của phần mềm vào/ra
- Các lớp phần mềm vào/ra
- Quản lý một số thiết bị thông dụng

5.1. Giới thiệu về quản lý vào/ra

- Vào/ra là quá trình trao đổi thông tin giữa bộ nhớ trong hoặc CPU với thiết bị vào/ra
- Nhiệm vụ của quản lý vào/ra được phụ trách bởi một phân hệ, phân hệ này thường nằm trong nhân của hệ điều hành.

5.1. Giới thiệu về quản lý vào/ra

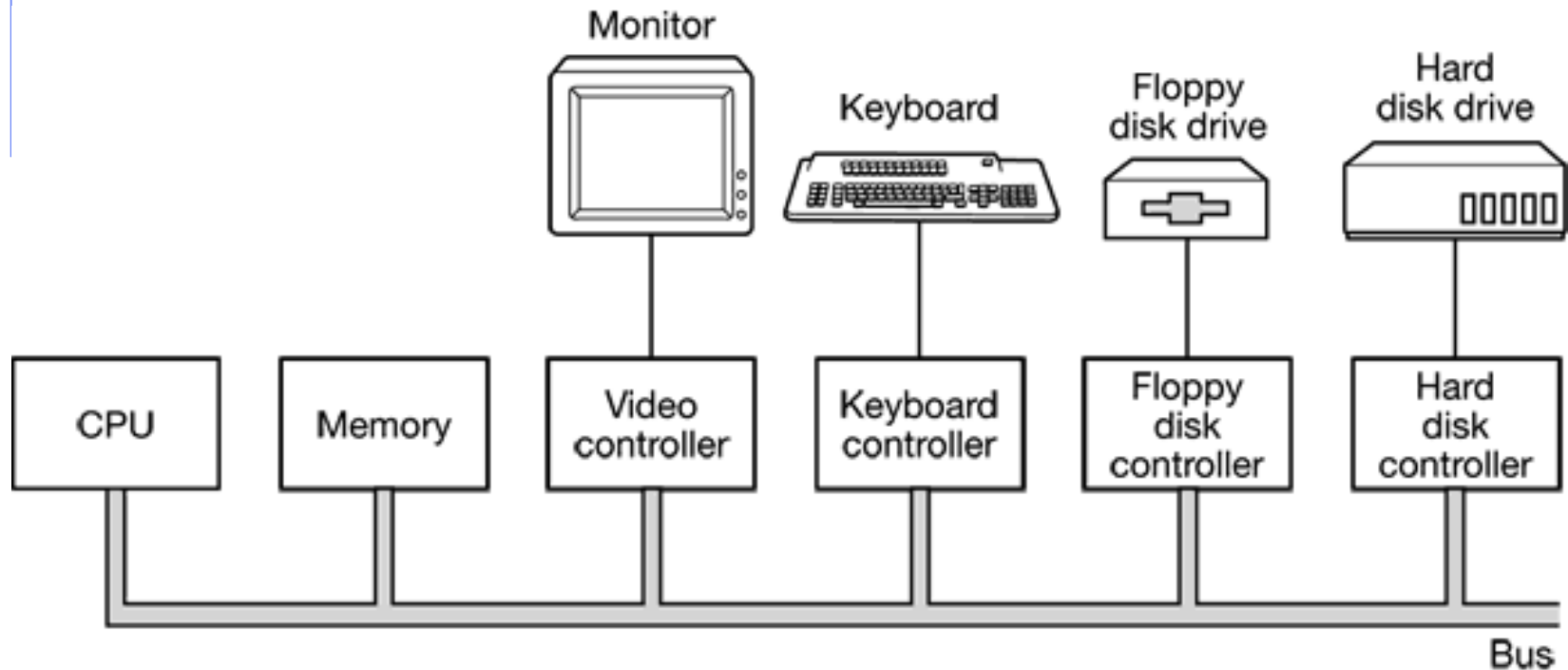
- Các yêu cầu đối với quản lý vào/ra:
 - Tạo ra giao diện chung và chuẩn cho phép làm việc với nhiều kiểu thiết bị mà không phải quan tâm tới đặc điểm cụ thể của thiết bị.
 - Có khả năng mở rộng.
 - Có biện pháp để nâng cao hiệu năng vào/ra.

5.2. Nguyên lý của phần cứng vào/ra

- Sơ đồ hệ thống vào/ra
- Phân loại thiết bị vào/ra
- Bộ điều khiển thiết bị (device controller)
- Các phương thức giao tiếp với controller
- Truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)
- Ngắt

Sơ đồ hệ thống vào/ra

- Các thiết bị vào/ra được kết nối với hệ thống thông qua các bộ điều khiển thiết bị (device controllers)



Phân loại thiết bị vào/ra

Các thiết bị vào/ra được phân thành hai loại:

- Các thiết bị khối (Block devices)
- Các thiết bị kí tự (Character devices)

Các thiết bị khối (Block devices)

- Thông tin được lưu trữ thành các khối có kích thước cố định (thường từ 512 - 65536 bytes) và mỗi khối có địa chỉ riêng.
- Có thể đọc hoặc ghi từng khối một cách độc lập với các khối còn lại
- Ví dụ: các ổ đĩa

Các thiết bị kí tự (Character devices)

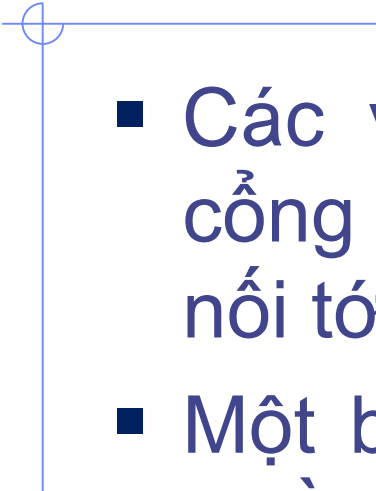
- Thiết bị ký tự là thiết bị có thể gửi hoặc nhận một chuỗi các ký tự mà không cần địa chỉ hoá và không cần thực hiện các hoạt động tìm kiếm các kí tự đó
- Ví dụ: Máy in, card mạng, chuột

5.2. Nguyên lý của phần cứng vào/ra

- Sơ đồ hệ thống vào/ra
- Phân loại thiết bị vào/ra
- **Bộ điều khiển thiết bị (device controller)**
- Các phương thức giao tiếp với controller
- Truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)
- Ngắt

Bộ điều khiển thiết bị (device controller)

- Các đơn vị vào/ra thường bao gồm thành phần cơ khí và thành phần điện tử
 - Thành phần cơ khí là bản thân thiết bị
 - Thành phần điện tử được gọi là **bộ điều khiển thiết bị (device controller)** hoặc **bộ điều hợp (adapter)**, nó thường có dạng một vỉ mạch (card) có thể cắm trên các khe mở rộng

- 
- Các vĩ mạch điều khiển thường có một cổng kết nối, từ đó cáp sẽ được gắn để nối tới thiết bị
 - Một bộ điều khiển có thể xử lý một hoặc nhiều thiết bị cùng loại
 - Người ta thường chuẩn hoá giao diện giữa bộ điều khiển và thiết bị để tăng khả năng tương thích giữa chúng
 - Ví dụ: các công ty khác nhau có thể sản xuất các ổ đĩa có cùng giao diện tiêu chuẩn IDE hoặc SCSI

- Nhiệm vụ của bộ điều khiển thiết bị là chuyển đổi dãy bit nối tiếp thành một khối nhiều byte và thực hiện việc sửa lỗi nếu cần
 - Đầu tiên, khối các byte này sẽ được lắp ghép lại từ nhiều bit, tại một vùng đệm bên trong bộ điều khiển
 - Sau khi phần checksum đã được kiểm tra, và không thấy có lỗi, nó có thể được sao chép vào bộ nhớ chính

5.2. Nguyên lý của phần cứng vào/ra

- Sơ đồ hệ thống vào/ra
- Phân loại thiết bị vào/ra
- Bộ điều khiển thiết bị (device controller)
- **Các phương thức giao tiếp với controller**
- Truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)
- Ngắt

Các phương thức giao tiếp với controller

- Mỗi bộ điều khiển thiết bị (controller) có một vài thanh ghi dùng để liên lạc với CPU
 - Bằng cách ghi dữ liệu vào các thanh ghi này, hệ điều hành có thể ra lệnh cho thiết bị gửi/nhận dữ liệu, bật/tắt thiết bị, ...
 - Nhờ đọc dữ liệu từ các thanh ghi này, hệ điều hành có thể biết được trạng thái của thiết bị, xem nó có sẵn sàng để nhận lệnh mới hay không
- Ngoài các thanh ghi điều khiển, nhiều thiết bị còn có bộ đệm dữ liệu mà HDH có thể đọc hoặc ghi

Các phương thức giao tiếp với controller (tiếp)

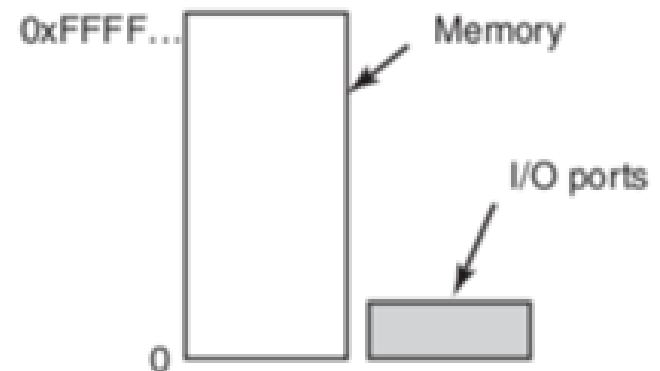
Làm cách nào mà CPU có thể liên lạc với các thanh ghi điều khiển đó?

- Cách 1: Dùng địa chỉ cổng
- Cách 2: Ánh xạ thanh ghi tới bộ nhớ

Cách 1: Dùng địa chỉ cổng

- Mỗi thanh ghi điều khiển sẽ được cấp một **địa chỉ cổng (I/O port number)**, đó là một số nguyên 8 hoặc 16 bit.
 - Nếu dùng địa chỉ cổng **8 bit** thì dải địa chỉ là: 00h -> FFh
 - Nếu dùng địa chỉ cổng **16 bit** thì dải địa chỉ là: 0000h -> FFFFh
- Không gian địa chỉ vào/ra là hoàn toàn tách biệt với không gian địa chỉ bộ nhớ.
- CPU dùng lệnh **IN** để đọc dữ liệu từ thanh ghi điều khiển và lệnh **OUT** để ghi dữ liệu ra thanh ghi điều khiển

Two address



Cách 1: Dùng địa chỉ cổng (tiếp)

- CPU có thể đọc dữ liệu từ thanh ghi điều khiển có địa chỉ PORT và lưu kết quả vào thanh ghi REG bên trong CPU bằng lệnh **IN**:

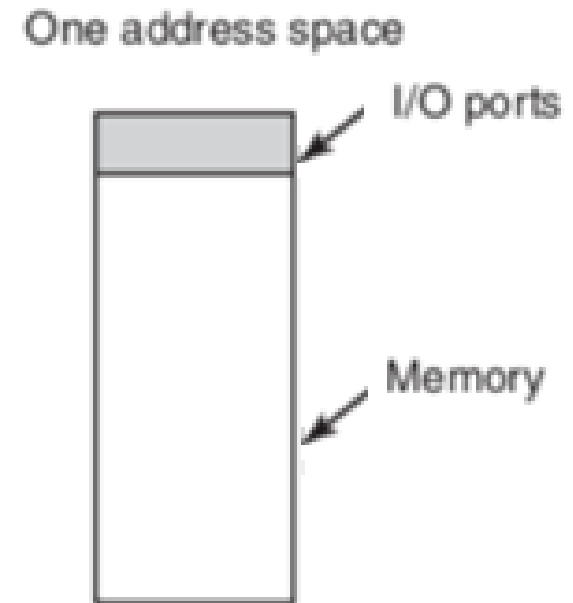
IN REG,PORT

- CPU có thể ghi dữ liệu từ thanh ghi REG của nó vào thanh ghi điều khiển bằng lệnh **OUT**:

OUT PORT,REG

Cách 2: Ánh xạ thanh ghi tới bộ nhớ

- Ánh xạ tất cả các thanh ghi điều khiển vào không gian bộ nhớ
 - Mỗi thanh ghi điều khiển được cấp một địa chỉ bộ nhớ duy nhất (địa chỉ này sẽ không dùng để cấp cho bộ nhớ)
- Truy nhập vào một thanh ghi điều khiển giống như truy nhập vào một ô nhớ thông thường
 - Thực hiện vào/ra bằng các lệnh truy nhập bộ nhớ thông thường



Ưu và nhược điểm của phương pháp ánh xạ thanh ghi tới bộ nhớ

■ Ưu điểm:

- Do giảm được số lệnh thực hiện nên sẽ làm tăng tốc độ vào/ra
- Tránh được xung đột giữa các tiến trình khi thực hiện vào/ra mà không cần áp dụng thêm các giải pháp bảo vệ khác (vì không gian nhớ của các tiến trình đã được hệ điều hành bảo vệ)

■ Nhược điểm:

- Cần được trang bị phần cứng chuyên dụng, thiết kế phần cứng sẽ trở nên phức tạp hơn

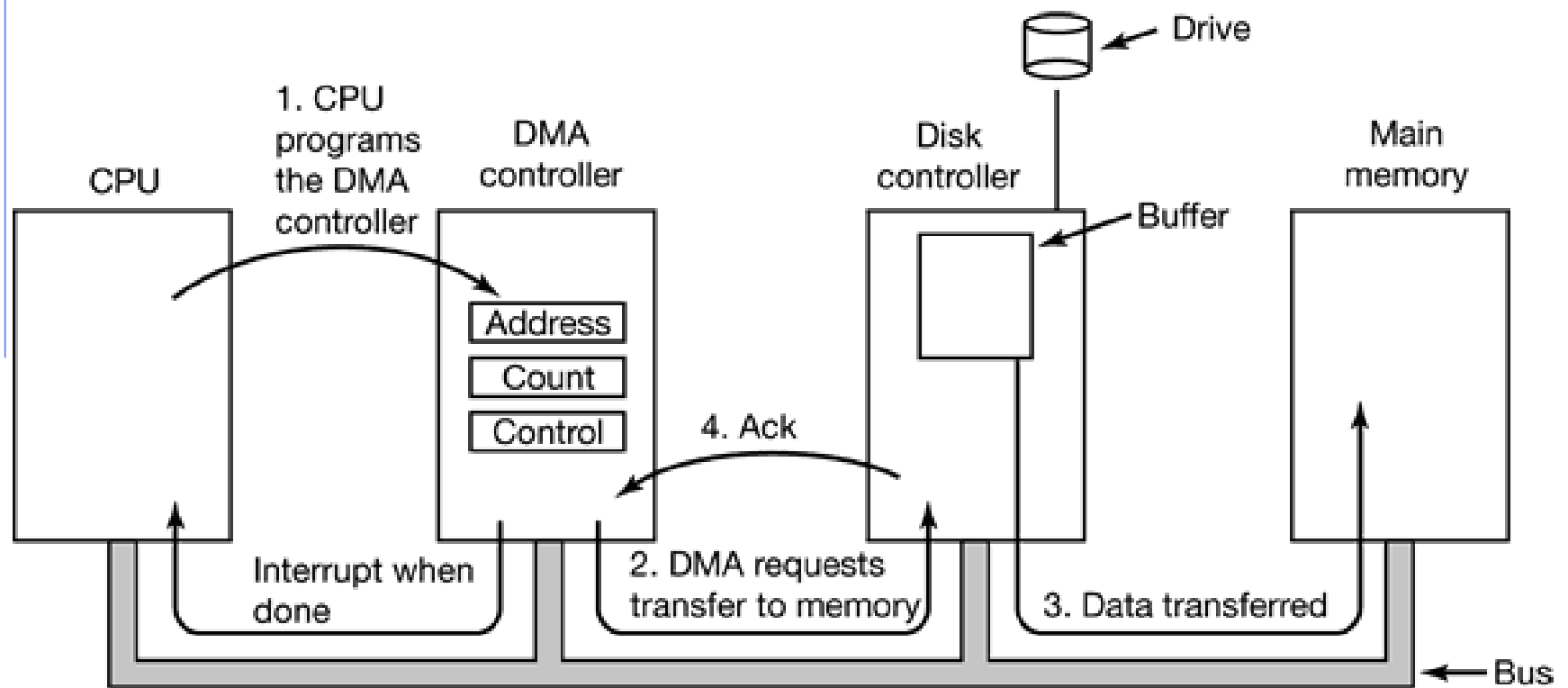
5.2. Nguyên lý của phần cứng vào/ra

- Sơ đồ hệ thống vào/ra
- Phân loại thiết bị vào/ra
- Bộ điều khiển thiết bị (device controller)
- Các phương thức giao tiếp với controller
- Truy nhập trực tiếp bộ nhớ (**Direct Memory Access - DMA**)
- Ngắt

Truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)

- CPU có thể yêu cầu dữ liệu từ các Controller theo từng byte một, điều này làm lãng phí thời gian của CPU.
- Cơ chế truy nhập trực tiếp bộ nhớ (Direct Memory Access –DMA) giúp tránh lãng phí thời gian CPU yêu cầu từng byte từ các controller
- Để sử dụng cơ chế DMA cần trang bị bộ điều khiển DMA (DMA controller)
 - DMA controller có thể được tích hợp vào các bộ điều khiển của từng thiết bị => mỗi thiết bị cần có DMA controller riêng
 - Phổ biến hơn, một DMA controller dùng chung cho nhiều thiết bị

Hoạt động của DMA



Quá trình đọc dữ liệu từ thiết bị vào bộ nhớ khi không có DMA

- Controller sẽ đọc một khối dữ liệu từ thiết bị vào vùng đệm bên trong controller
- Sau khi kiểm tra lỗi, controller sẽ phát sinh một ngắt gửi tới CPU để thông báo
- CPU sẽ thực hiện lệnh để đọc dữ liệu từ vùng đệm trên controller vào bộ nhớ.

Như vậy controller sẽ điều khiển quá trình đọc dữ liệu từ thiết bị vào bộ đệm của nó, còn CPU sẽ điều khiển quá trình đọc dữ liệu từ bộ đệm của controller vào bộ nhớ.

Quá trình đọc dữ liệu từ thiết bị vào bộ nhớ bằng DMA

■ *Bước 1:*

- CPU sẽ lập trình cho DMA controller bằng cách đặt thông tin vào các thanh ghi trong nó, để nó biết được đối tượng sẽ trao đổi dữ liệu.
- CPU cũng gửi một lệnh tới bộ điều khiển đĩa, yêu cầu đọc dữ liệu từ đĩa và cất vào bộ đệm trong bộ điều khiển, rồi kiểm tra checksum.

Quá trình đọc dữ liệu từ thiết bị vào bộ nhớ bằng DMA (tiếp)

- *Bước 2: Khi đã có dữ liệu hợp lệ trong bộ đệm của controller.*
 - DMA controller gửi một yêu cầu đọc dữ liệu qua bus tới bộ điều khiển đĩa.
- *Bước 3:*
 - Dữ liệu từ bộ đệm của controller được ghi vào bộ nhớ

Quá trình đọc dữ liệu từ thiết bị vào bộ nhớ bằng DMA (tiếp)

■ *Bước 4:*

- Khi quá trình ghi hoàn thành, bộ điều khiển đĩa sẽ gửi một tín hiệu ACK (báo hiệu thành công) qua bus tới DMA controller. DMA sẽ tăng địa chỉ của ô nhớ và giảm bộ đếm số byte
- Lặp lại bước 2 đến 4 cho tới khi bộ đếm số byte bằng 0
- DMA controller sẽ gửi tín hiệu ngắt tới CPU để thông báo quá trình truyền dữ liệu đã hoàn tất.

Quá trình đọc dữ liệu từ thiết bị vào bộ nhớ bằng DMA (tiếp)

- Controller sẽ điều khiển quá trình đọc dữ liệu từ thiết bị vào bộ đệm của nó
- Bộ điều khiển DMA sẽ điều khiển quá trình đọc dữ liệu từ bộ đệm của controller vào bộ nhớ

Như vậy, nhờ có bộ điều khiển DMA, CPU sẽ không phải tốn nhiều thời gian cho việc thực hiện vào/ra nữa

Nhược điểm của việc sử dụng DMA

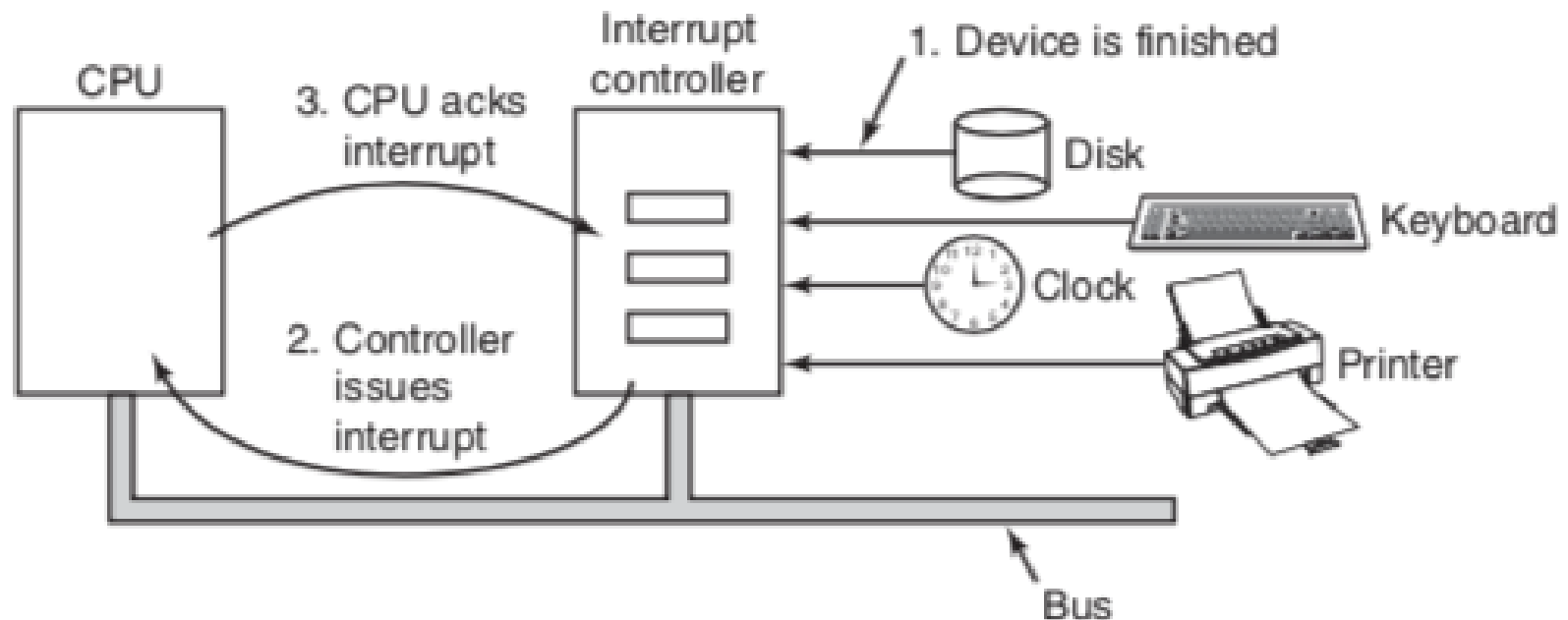
- Giá thành phần cứng sẽ tăng lên
- Nếu CPU không có công việc gì để làm, thì việc bắt CPU (có tốc độ cao) phải ngồi chờ DMA controller (có tốc độ thấp) thực hiện vào/ra là điều bất hợp lý
- Do đó không phải tất cả các máy tính đều sử dụng DMA, nhất là các dòng máy tính giá rẻ

5.2. Nguyên lý của phần cứng vào/ra

- Sơ đồ hệ thống vào/ra
- Phân loại thiết bị vào/ra
- Bộ điều khiển thiết bị (device controller)
- Các phương thức giao tiếp với controller
- Truy nhập trực tiếp bộ nhớ (Direct Memory Access - DMA)
- Ngắt

Ngắt (interrupt)

- Khi kết thúc công việc, thiết bị vào/ra sẽ gửi tín hiệu ngắt lên bus – chip điều khiển ngắt trên bo mạch chủ sẽ phát hiện và xử lý ngắt.



Chương 5: Quản lý vào ra

- Giới thiệu về quản lý vào/ra
- Nguyên lý của phần cứng vào/ra
- **Nguyên lý của phần mềm vào/ra**
- Các lớp phần mềm vào/ra
- Quản lý một số thiết bị thông dụng

5.3. Nguyên lý của phần mềm vào/ra

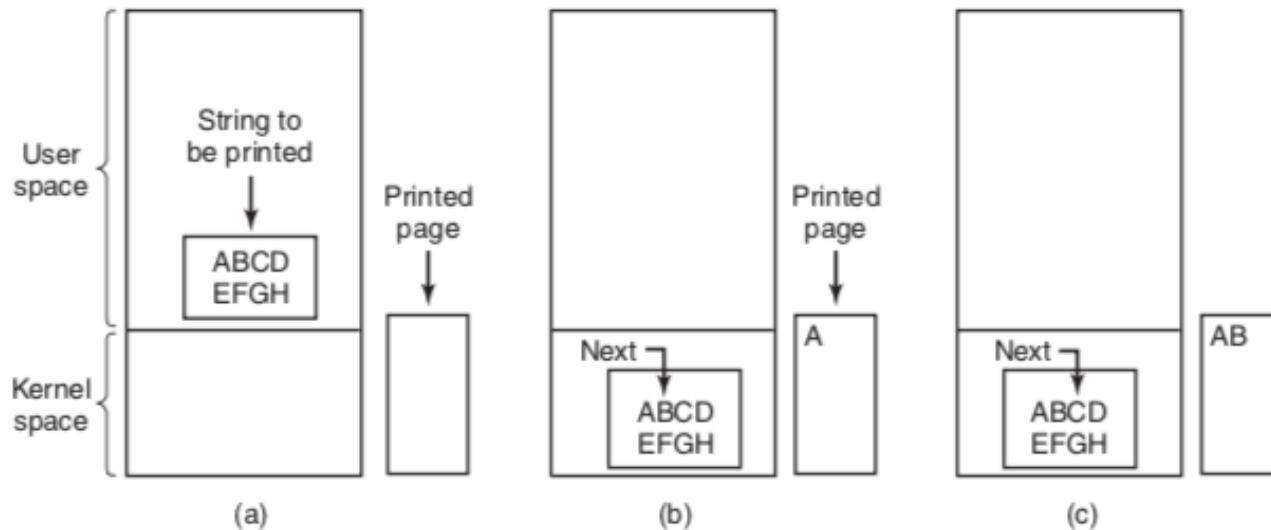
- Mục đích của phần mềm vào/ra
- Vào/ra theo chương trình
- Vào/ra bằng ngắt
- Vào/ra bằng DMA

Mục đích của phần mềm vào/ra

- **Đảm bảo Độc lập thiết bị:** Viết chương trình vào/ra có thể truy cập bất cứ thiết bị vào/ra nào mà không cần phải biết trước thiết bị.
- **Đặt tên đồng nhất:** tất cả các tệp và thiết bị chung nhau quy tắc đặt tên không phụ thuộc vào loại thiết bị và được xác định bởi đường dẫn.
- **Xử lý lỗi:** nên được thực thi ở phần cứng hoặc tại trình điều khiển thiết bị.
- **Lưu đệm (buffer):** tăng hiệu năng vào/ra.
-

Vào/ra theo chương trình (Programmed I/O)

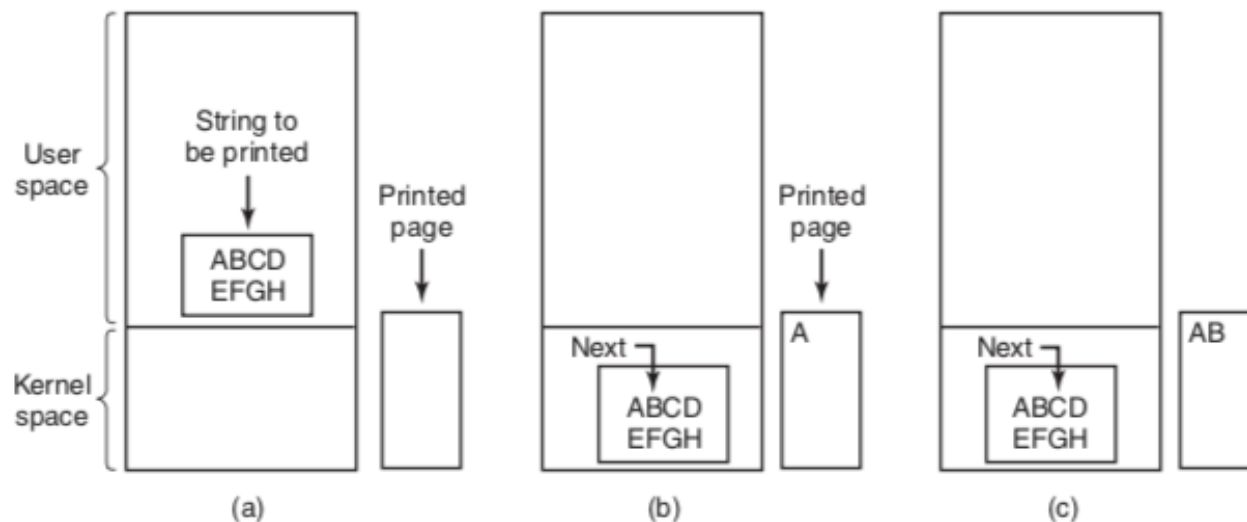
- Cách đơn giản nhất để vào/ra là để CPU làm mọi việc, gọi là vào/ra bằng chương trình
- Ví dụ:
 - Xét một tiến trình của người dùng muốn gửi một chuỗi gồm 8 ký tự “ABCDEFGH” ra máy in



Hình 5-6 Các bước in một chuỗi kí tự.

Vào/ra theo chương trình (Programmed I/O)

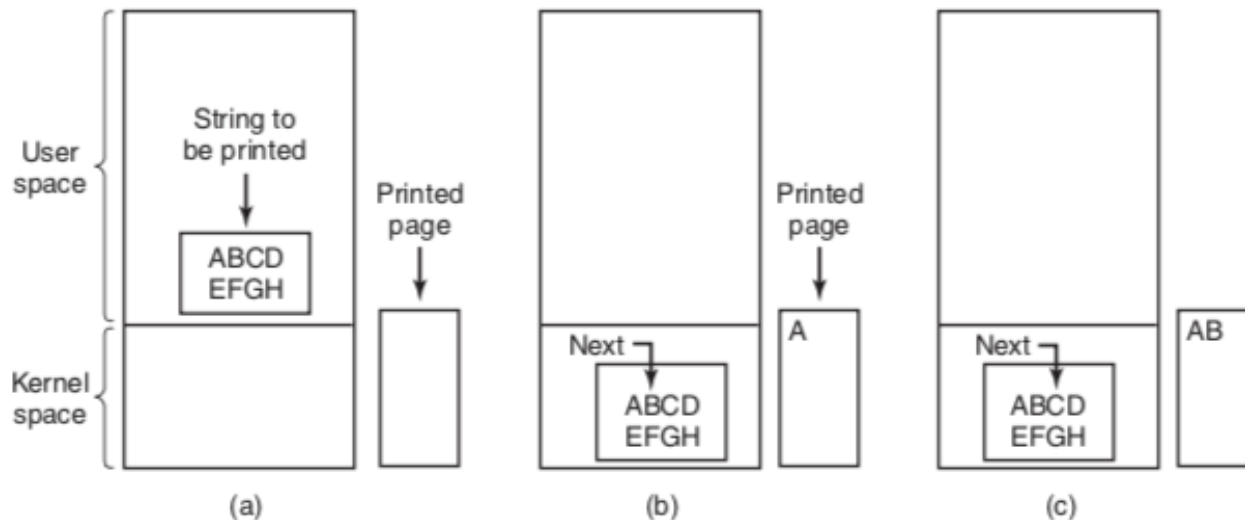
- Ví dụ (tiếp)



- Đặt chuỗi kí tự vào bộ đệm trong không gian người dùng (Hình 5-6(a))
- Tiến trình người dùng sẽ tạo một lời gọi hệ thống để giành quyền sử dụng máy in.

Vào/ra theo chương trình (Programmed I/O)

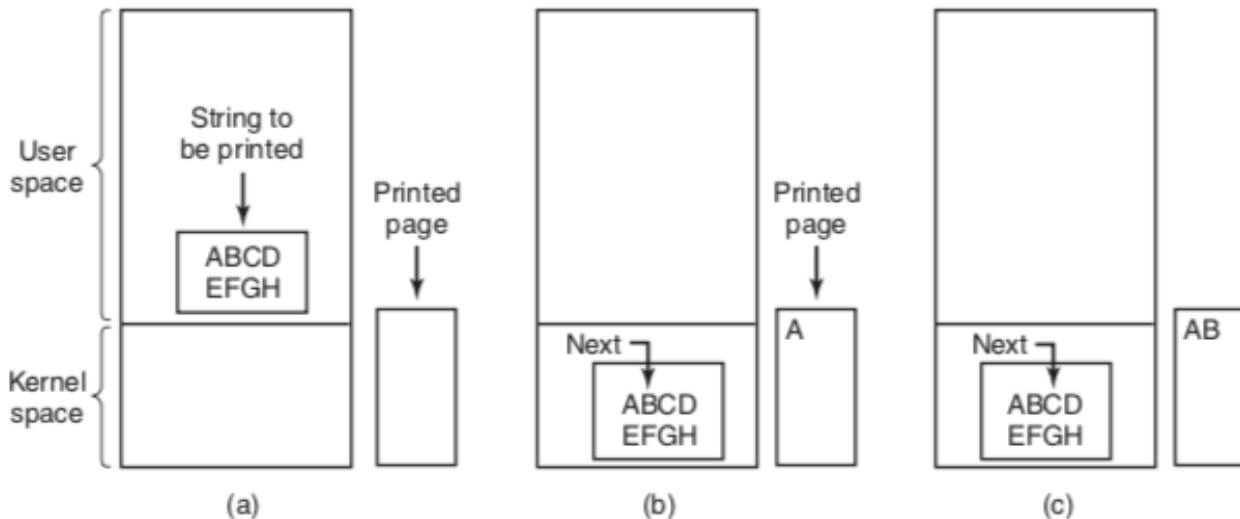
- Ví dụ (tiếp)



- HĐH sẽ sao chép bộ đệm đang chứa chuỗi kí tự (tạm gọi là p) vào không gian kernel

Vào/ra theo chương trình (Programmed I/O)

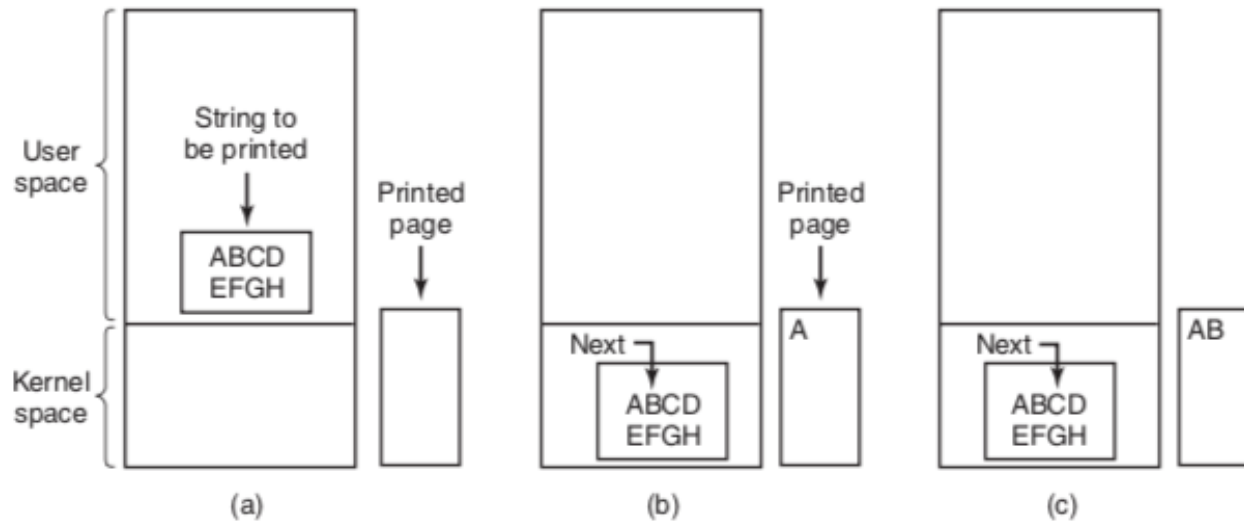
■ Ví dụ (tiếp)



- HĐH kiểm tra xem máy in đã sẵn sàng chưa.
 - ◆ Nếu chưa thì nó sẽ chờ.
 - ◆ Nếu sẵn sàng, HĐH sẽ sao chép kí tự đầu tiên vào thanh ghi dữ liệu của máy in và kích hoạt máy in (kí tự đầu tiên A được in) (Hình 5-6 (b)) và đánh dấu kí tự "B" là kí tự sẽ được in tiếp theo

Vào/ra theo chương trình (Programmed I/O)

■ Ví dụ (tiếp)



- HĐH sẽ chờ cho tới khi máy in sẵn sàng trở lại. Khi đó kí tự tiếp theo (B) sẽ được in (Hình 5-6(c))
- Vòng lặp tiếp tục cho tới khi toàn bộ chuỗi được in hết.
- Sau đó quyền điều khiển được trả về cho tiến trình của người dùng.

Vào/ra theo chương trình (Programmed I/O)

- Các hành động nói trên của HĐH được tóm tắt như hình sau:

```
copy_from_user(buffer, p, count);          /* p là bộ đệm của kernel */
for (i = 0; i < count; i++) {               /* Lặp với tất cả các kí tự */
    while (*printer_status_reg != READY) ; /* Lặp cho tới khi sẵn sàng */
    *printer_data_register = p[i];          /* Kết xuất một kí tự */
}
return_to_user();
```

- Đầu tiên dữ liệu được sao chép vào kernel
- Sau đó hệ điều hành khởi tạo một vòng lặp để kết xuất từng kí tự:
 - ♦ Cứ mỗi khi kết xuất được một kí tự thì CPU lại tiếp tục thăm dò thiết bị xem nó có sẵn sàng nhận kí tự tiếp theo không
- Sau khi chuỗi ký tự được in hết, chuyển quyền điều khiển cho tiến trình của người dùng.

Vào/ra bằng ngắt (Interrupt-Driven I/O)

- Một giải pháp để CPU có thể làm một công việc gì đó (trong lúc chờ máy in đạt trạng thái sẵn sàng) là sử dụng các ngắt.

```
copy_from_user(buffer, p, count);
enable_interrupts();
while(*printer_status_reg != READY)
;
*printer_data_register = p[0];
scheduler();
```

(a)

```
if(count == 0) {
    unblock_user();
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```

(b)

Hình 5-8: Ghi một chuỗi ký tự ra máy in bằng phương pháp vào/ra bằng ngắt. (a) Mã thi hành khi có hàm hệ thống. (b) Thủ tục xử lý ngắt.

Vào/ra bằng ngắt (Interrupt-Driven I/O)

```
copy_from_user(buffer, p, count);
enable_interrupts();
while(*printer_status_reg != READY)
;
*printer_data_register = p[0];
scheduler();
```

(a)

```
if(count == 0) {
    unblock_user();
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```

(b)

- Khi có hàm hệ thống yêu cầu in chuỗi, bộ đệm được sao chép vào không gian kernel
- Kí tự đầu tiên được sao chép tới máy in ngay khi máy in chấp nhận. Lúc này CPU sẽ gọi bộ phận điều độ, và một tiến trình nào đó sẽ được chạy (Hình 5-8 (a))

Vào/ra bằng ngắt (Interrupt-Driven I/O)

```
copy_from_user(buffer, p, count);  
enable_interrupts();  
while(*printer_status_reg != READY)  
;  
*printer_data_register = p[0];  
scheduler();
```

(a)

```
if(count == 0) {  
    unblock_user();  
} else {  
    *printer_data_register = p[i];  
    count = count - 1;  
    i = i + 1;  
}  
acknowledge_interrupt();  
return_from_interrupt();
```

(b)

- Khi máy in in xong một kí tự và chuẩn bị nhận kí tự tiếp theo, nó sẽ phát ra một tín hiệu ngắt.
- Ngắt này sẽ dừng tiến trình hiện hành và lưu lại trạng thái của tiến trình đó.
- Sau đó thủ tục xử lý ngắt máy in được chạy. Một phiên bản đơn giản của thủ tục này được trình bày ở hình 5-8(b).
- Nếu không có kí tự nào cần in nữa, thủ tục xử lý ngắt sẽ đánh thức tiến trình người dùng đang bị dừng. Còn nếu vẫn có kí tự cần in, nó sẽ gửi kí tự tiếp theo ra máy in, thông báo hoàn thành ngắt, và trở về tiến trình hiện hành (là tiến trình đang chạy khi xảy ra ngắt), tiến trình này lại tiếp tục thực hiện công việc của nó.

Vào/ra bằng DMA (I/O using DMA)

- Bộ điều khiển DMA (DMA Controller) thực hiện việc gửi kí tự ra máy in, chứ không dùng đến CPU

```
copy_from_user(buffer, p, count);  
set_up_DMA_controller();  
scheduler();
```

(a)

```
acknowledge_interrupt();  
unblock_user();  
return_from_interrupt();
```

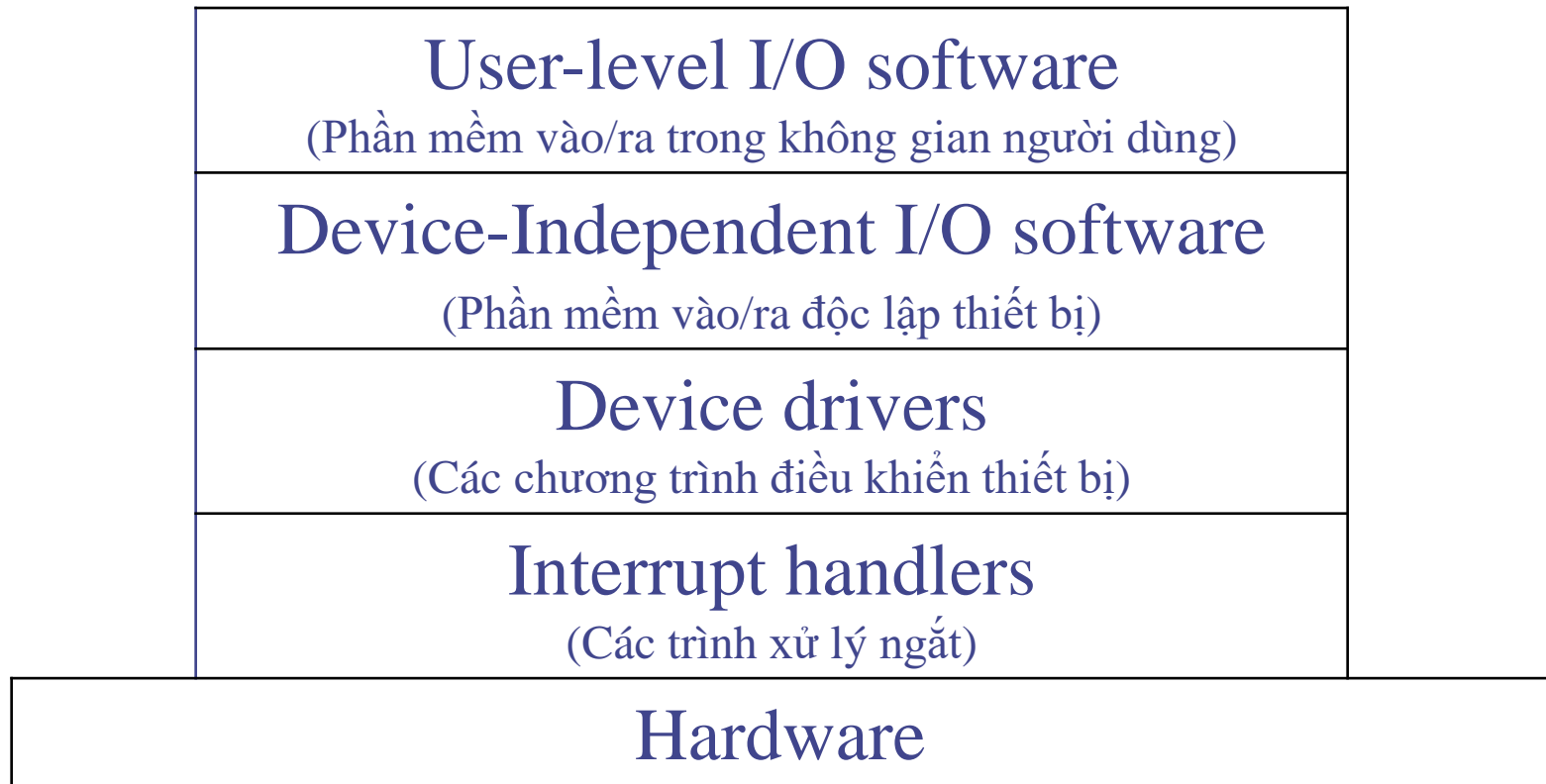
(b)

Hình 5-9: In một chuỗi ký tự bằng DMA. (a) Mã thi hành khi có hàm hệ thống.
(b) Thủ tục xử lý ngắt.

Chương 5: Quản lý vào ra

- Giới thiệu về quản lý vào/ra
- Nguyên lý của phần cứng vào/ra
- Nguyên lý của phần mềm vào/ra
- **Các lớp phần mềm vào/ra**
- Quản lý một số thiết bị thông dụng

5.4. Các lớp phần mềm vào/ra



Các trình xử lý ngắt (Interrupt handlers)

- Là các chương trình giao tiếp với phần cứng ở cấp thấp nhất
- Một số các bước phải thực hiện bằng phần mềm khi có ngắt cứng xảy ra:
 1. Lưu trạng thái các thanh ghi
 2. Cài đặt ngưỡng cảnh cho thủ tục của dịch vụ ngắt (MMU và bảng trang)
 3. Cài đặt stack cho thủ tục dịch vụ ngắt
 4. Báo cho khối điều khiển ngắt
 5. Sao chép nội dung thanh ghi vào bảng tiến trình

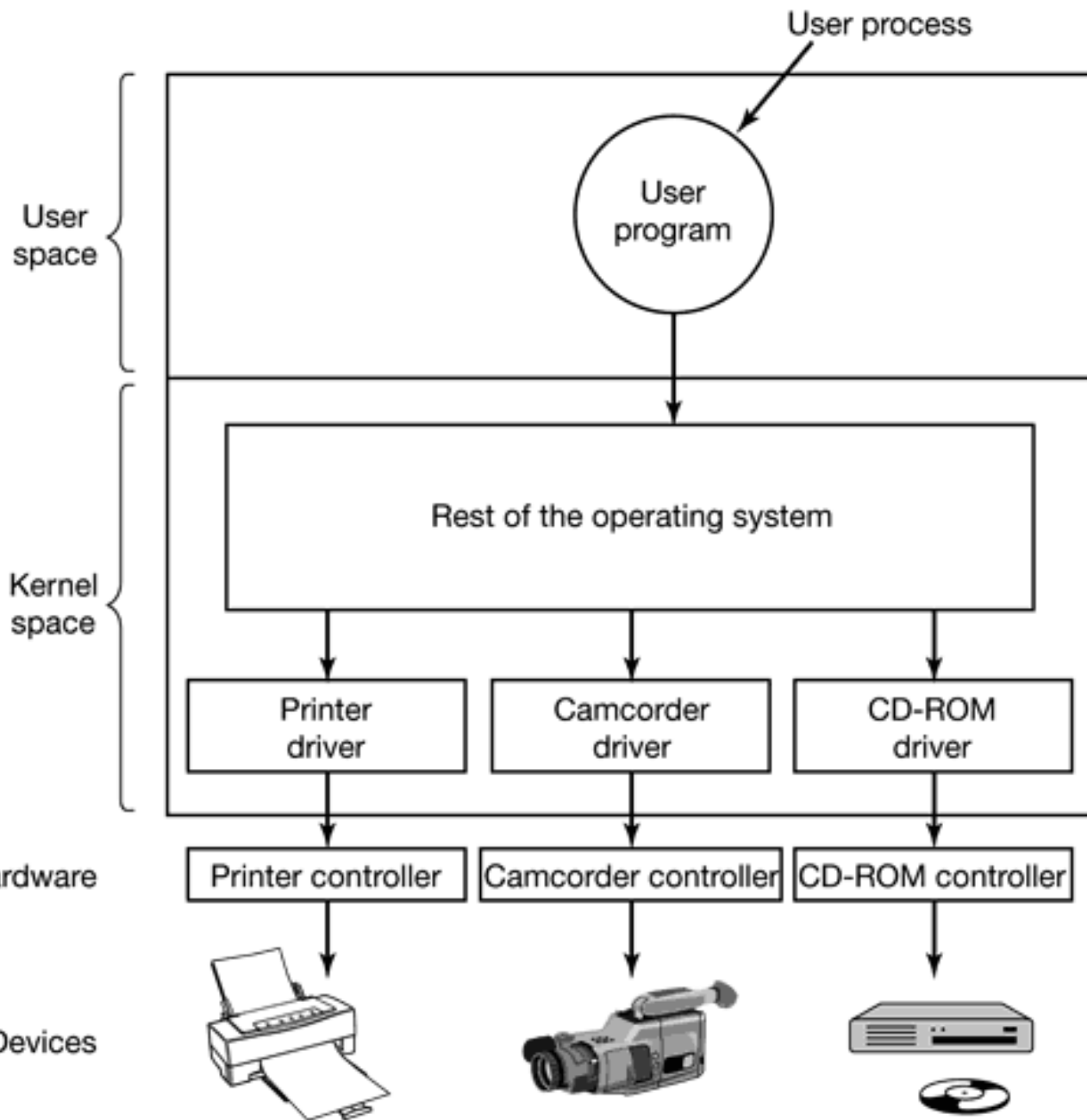
Các trình xử lý ngắt (Interrupt handlers)

- Một số các bước phải thực hiện bằng phần mềm khi có ngắt cứng xảy ra (tiếp):
 6. Chạy thủ tục dịch vụ ngắt. Lấy ra thông tin từ thanh ghi khối điều khiển ngắt
 7. Lựa chọn tiến trình chạy tiếp theo
 8. Cài đặt ngưỡng cảnh MMU cho tiến trình chạy tiếp theo
 9. Tải thanh ghi tiến trình mới
 10. Tiến trình mới bắt đầu

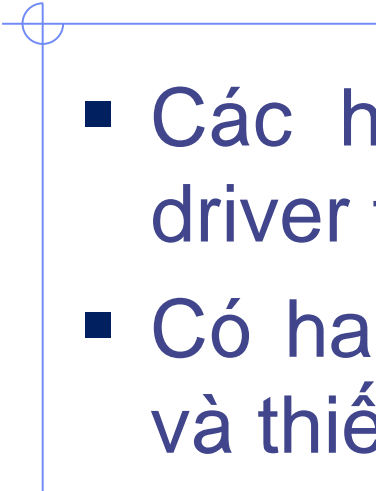
Các chương trình điều khiển thiết bị (Device Drivers)

- Mỗi thiết bị vào/ra kết nối với máy tính đều cần tới các mã lệnh đặc trưng cho thiết bị để điều khiển nó. Các mã này được gọi là **trình điều khiển thiết bị (device driver)**
- Trình điều khiển thiết bị thường được viết bởi nhà sản xuất và được phân phối kèm theo thiết bị.
- Các hệ điều hành thường tích hợp sẵn một số trình điều khiển thiết bị bên trong nó

- Mỗi trình điều khiển thiết bị thường chỉ dùng để điều khiển cho một loại thiết bị, hoặc một lớp các thiết bị có quan hệ gần gũi với nhau
- ***Ví dụ:*** Một trình điều khiển đĩa SCSI thường điều khiển được nhiều loại đĩa SCSI, với các dung lượng và tốc độ khác nhau



- Các driver (được cung cấp bởi nhà sản xuất thiết bị) sẽ được cài đặt vào hệ điều hành.
- Hệ điều hành cần phải có một kiến trúc cho phép việc thực hiện cài đặt các drivers

- 
- Các hệ điều hành thường phân loại các driver theo loại của thiết bị
 - Có hai loại thiết bị phổ biến là thiết bị khối và thiết bị kí tự, do đó cũng có hai loại driver tương ứng là driver khối và driver kí tự

Phần mềm vào/ra độc lập thiết bị (Device-Independent I/O Software)

- Chức năng cơ bản của phần mềm độc lập thiết bị:
 - Thực hiện các chức năng vào/ra chung cho mọi thiết bị
 - Cung cấp một giao diện đồng nhất cho các phần mềm cấp người dùng.

Phần mềm vào/ra không gian người dùng (User-level I/O software)

- Hàm hệ thống cho việc vào/ra, thường được tạo ra bởi các thủ tục trong thư viện. Tập hợp tất cả các thủ tục thư viện này là một phần của hệ thống vào/ra.

Ví dụ trong C:

```
printf("Gia tri cua a = %d", a);
```

Chương 5: Quản lý vào ra

- Giới thiệu về quản lý vào/ra
- Nguyên lý của phần cứng vào/ra
- Nguyên lý của phần mềm vào/ra
- Các lớp phần mềm vào/ra
- **Quản lý một số thiết bị thông dụng**

5.5. Quản lý một số thiết bị thông dụng

- Quản lý màn hình
- Quản lý đĩa từ

Quản lý màn hình

- Thông tin hiển thị trên màn hình được điều khiển bởi vi mạch màn hình (Video Adapter)
- Video Adapter có hai thành phần chính:
 - + *Bộ nhớ hiển thị (Display Memory)*
 - + *Mạch điều khiển màn hình (Video controller)*

Bộ nhớ hiển thị (Display Memory)

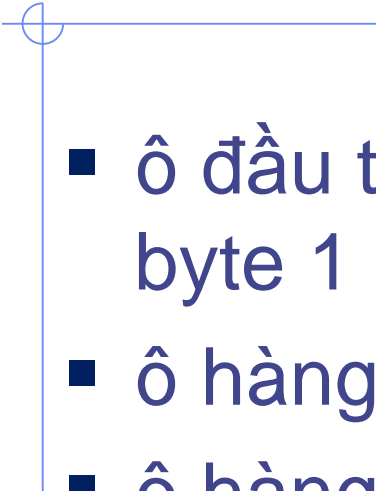
- Đây là nơi chứa những thông tin sẽ được hiện ra trên màn hình.
- CPU có thể liên lạc với Display Memory thông qua hệ thống bus địa chỉ, do đó có thể tác động tới vùng nhớ này bằng các lệnh truy nhập bộ nhớ thông thường

Mạch điều khiển màn hình (Video controller)

- Đây là trung tâm điều khiển quá trình hiện thông tin ra màn hình, nó có chứa một số thanh ghi, mỗi thanh ghi ứng với một địa chỉ cổng
 - Các địa chỉ cổng thường nằm trong dải từ **3B0h** đến **3DFh**
- Video controller có 2 chế độ hoạt động: Chế độ văn bản và chế độ đồ họa

Chế độ văn bản (text mode)

- Màn hình được chia thành nhiều ô, mỗi ô hiện 1 kí tự
- Chia thành 80 cột x 25 dòng, tổng số ô trên một màn hình là 2000 ô.
- Mỗi ô trên màn hình tương ứng với 2 byte của bộ nhớ hiển thị (2 byte này chứa gì thì trên màn hình hiện cái đó): Byte thấp chứa mã ASCII, byte cao chứa màu sắc.

- 
- ô đầu tiên (hàng 1, cột 1) ứng với byte 0 và byte 1 của bộ nhớ hiển thị
 - ô hàng 1 cột 2 ứng với byte 2 và byte 3
 - ô hàng 1 cột 3 ứng với byte 4 và byte 5
 - ô hàng 1 cột 4 ứng với byte 6 và byte 7
 - ...
 - ô hàng 25 cột 80 ứng với byte 3998 và 3999

Chế độ đồ họa (Graphic mode)

- Màn hình là một tập hợp các điểm sáng (pixel), mỗi điểm có một màu sắc riêng.
- Các điểm sáng phối hợp với nhau sẽ tạo ra các hình ảnh mong muốn
- Có nhiều chế độ màu sắc khác nhau

Chế độ đồ họa 256 màu

- Mỗi điểm sáng trên màn hình ứng với 1 byte trong bộ nhớ hiển thị

Chế độ đồ họa 16 triệu màu (24 bit)

- Mỗi điểm sáng trên màn hình ứng với 3 byte trong bộ nhớ hiển thị

R	G	B
0 - 255	0 - 255	0 - 255

5.5. Quản lý một số thiết bị thông dụng

- Quản lý màn hình
- **Quản lý đĩa từ**

Quản lý đĩa từ

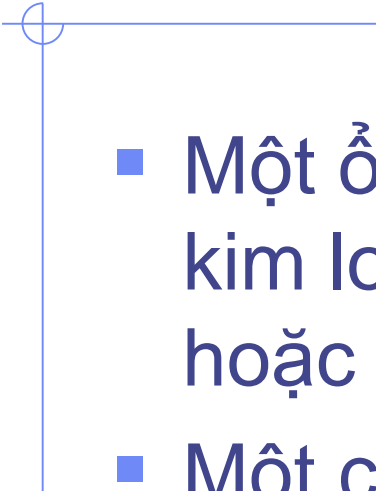
- Cấu trúc đĩa từ
- Đọc ghi đĩa từ
- Định dạng đĩa từ
- Các thuật toán lập lịch cánh tay đĩa

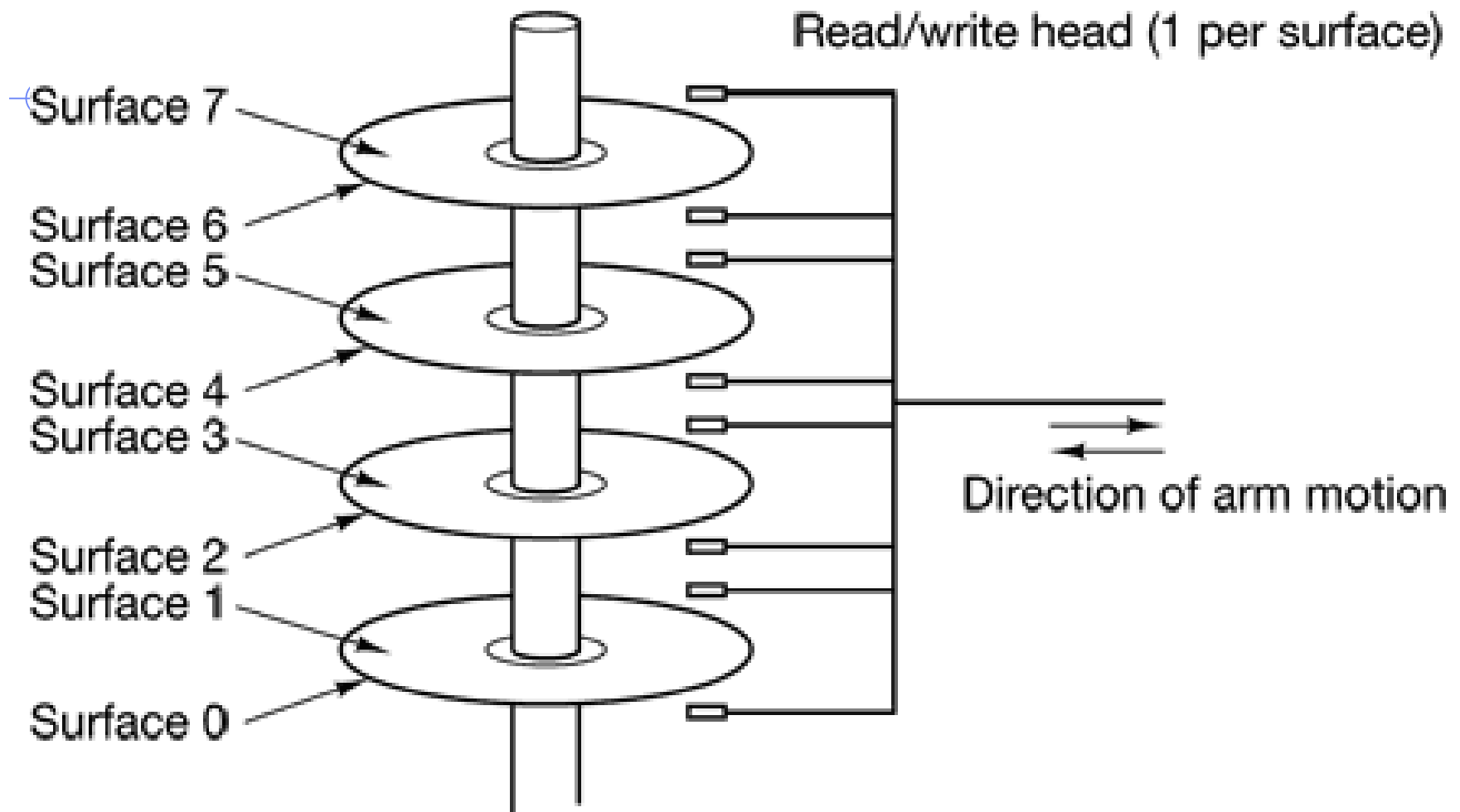
Cấu trúc đĩa từ

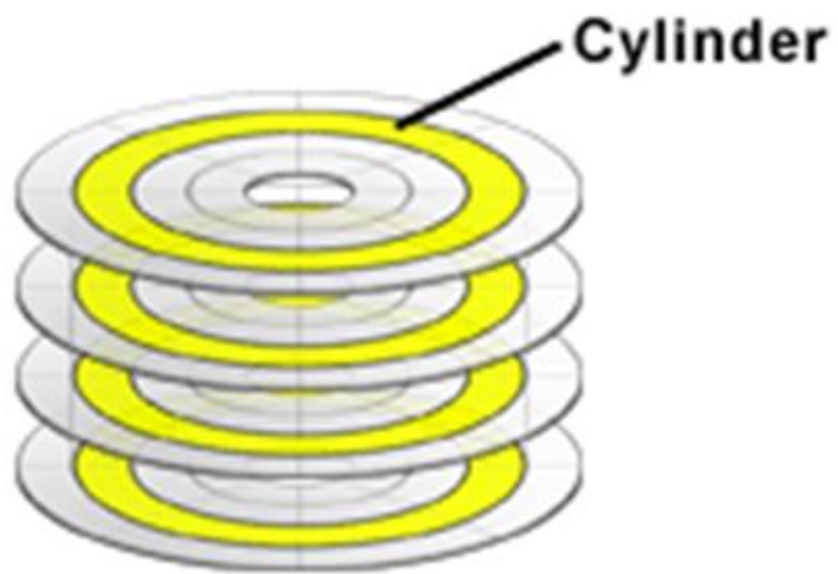
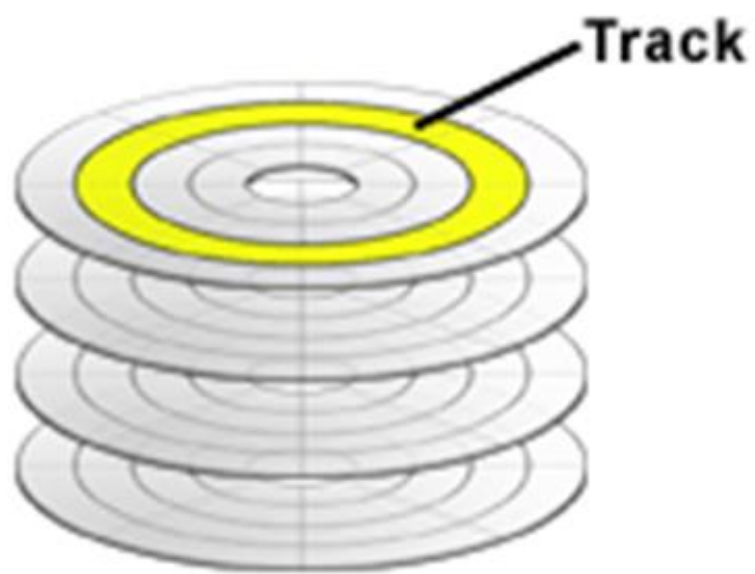
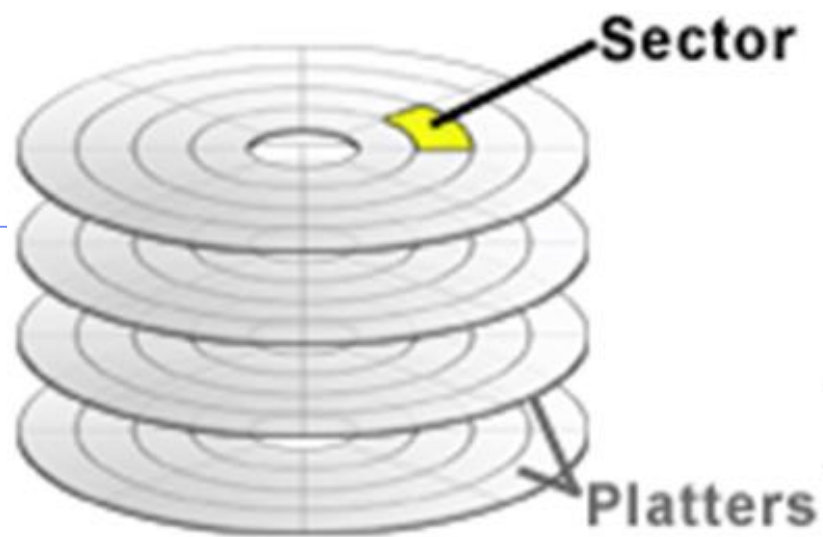
- Đĩa từ là thiết bị có dung lượng lưu trữ lớn, tốc độ đọc/ghi nhanh, là thiết bị lưu trữ chính của máy tính.





- 
- Một ổ đĩa có thể chứa một hoặc nhiều đĩa kim loại, với tốc độ quay là 5400, 7200, hoặc 10800 vòng/phút.
 - Một cánh tay cơ khí được gắn ở góc để đầu đọc/ghi (head) có thể chuyển động trên các bề mặt đĩa



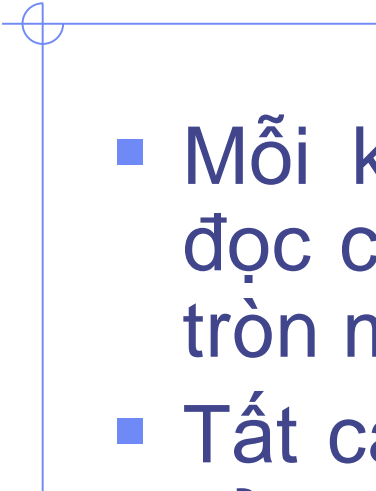


Để xác định 1 sector cần 3 thông số

- Số hiệu bề mặt (side)
- Số hiệu rãnh (track)
- Số hiệu cung (sector)

Hoặc:

- Số hiệu Head
- Số hiệu Cylinder
- Số hiệu sector

- 
- Mỗi khi cánh tay dịch chuyển, các đầu đọc có thể đọc được dữ liệu ở một vòng tròn mới, gọi là **rãnh (track)**
 - Tất cả các rãnh ứng với cùng một vị trí của cánh tay tạo thành một **trụ (cylinder)**
 - Mỗi rãnh được chia thành nhiều **cung từ (sector)**, thường có 512 byte trên mỗi cung từ.

- Thời gian chuyển động của cánh tay giữa hai trụ kế tiếp vào khoảng 1ms. Di chuyển cánh tay tới một trụ bất kỳ mất từ 5 tới 10 ms, tùy từng thiết bị.
- Khi cánh tay được đưa tới vị trí của rãnh, thiết bị sẽ phải chờ để cung từ quay tới vị trí đầu đọc, thời gian chờ khoảng 5 đến 10 ms, tùy vào tốc độ quay của đĩa.
- Đầu đọc sẽ thực hiện đọc (hoặc ghi) dữ liệu lên cung từ với tốc độ từ 5 MB/s tới 160 MB/s (tùy loại đĩa).

Định dạng đĩa từ

Đĩa mới sản xuất ra thì chưa sử dụng được ngay, cần trải qua 3 công đoạn sau:

- + Format cấp thấp
- + Phân khu đĩa
- + Format cấp cao

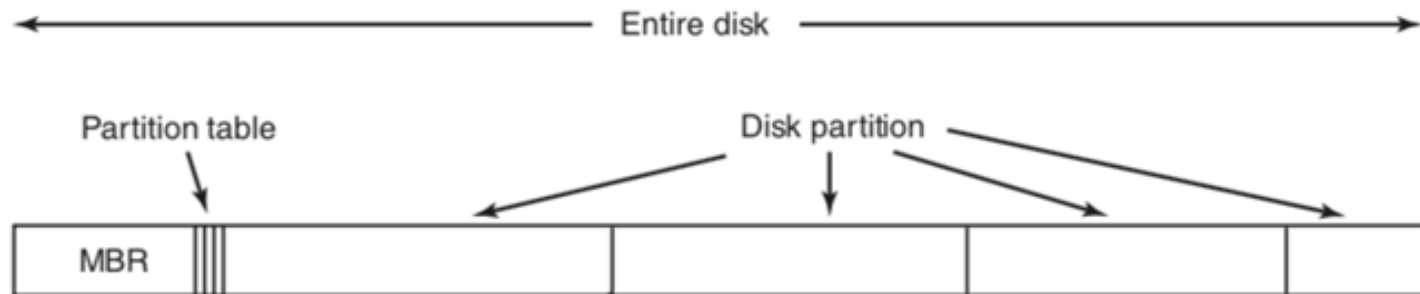
Format cấp thấp

- Tạo ra các track, các sector ở trên bề mặt đĩa bằng cách ghi các thông tin đánh dấu vị trí lên bề mặt đĩa.
- Định dạng của một sector:

Preamble	Data	ECC
----------	------	-----

- Phần mào đầu (Preamble): cho phép phần cứng nhận ra điểm bắt đầu của sector, số hiệu sector, số hiệu cylinder và một số thông tin khác.
- Trường mã sửa lỗi (ECC: error-correcting Code) chứa các thông tin thêm sử dụng để khôi phục khi có lỗi.

Phân khu đĩa



- Chia đĩa thành các phân khu. Về mặt logic, mỗi phân khu giống như một đĩa riêng rẽ.
- Vị trí bắt đầu và kết thúc của các phân khu được lưu vào bảng phân khu (partition table).

Format cấp cao

- Tạo ra các cấu trúc quản lý file trên từng phân khu.
- Việc format cấp cao sẽ tùy thuộc vào loại hệ điều hành mà ta sử dụng trên phân khu đó.

Các thuật toán lập lịch cánh tay đĩa

- Thời gian để đọc/ghi một khối dữ liệu của đĩa quyết định bởi 3 yếu tố:
 - Thời gian tìm kiếm (di chuyển cánh tay đĩa để định vị head tới cylinder cần tìm)
 - Độ trễ quay của đĩa
 - Thời gian truyền dữ liệu thực tế
- Trong đó, yếu tố thời gian tìm kiếm là quan trọng nhất và cần các thuật toán lập lịch để giảm thời gian tìm kiếm trung bình.

Các thuật toán lập lịch cánh tay đĩa

- Lập lịch cánh tay đĩa là xây dựng các thuật toán dịch chuyển đầu từ đọc/ghi (head) sao cho thời gian tìm kiếm trung bình là tối ưu nhất.
- Một số thuật toán:
 - FCFS (First Come First Served)
 - SSF (Shortest Seek First)
 - Thuật toán thang máy (elevator algorithm)

Ví dụ

Giả thiết

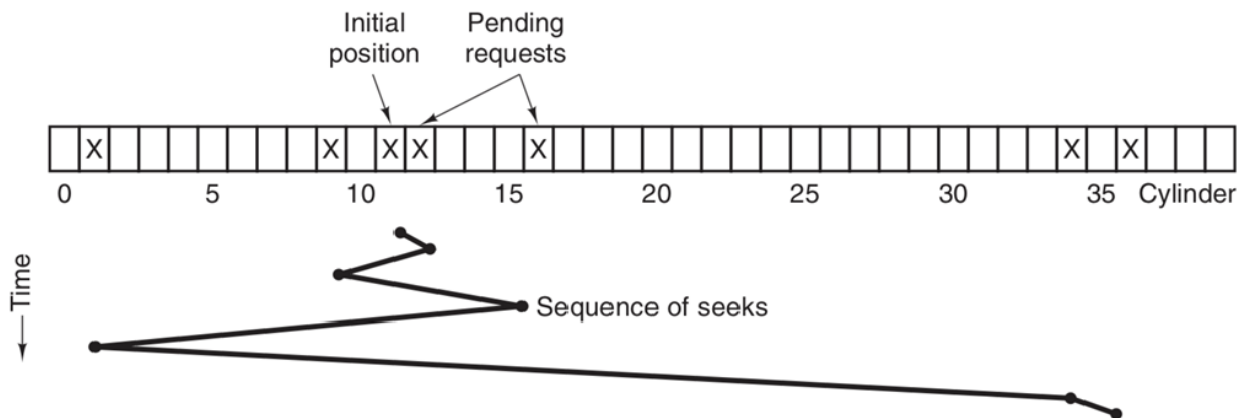
- Đầu đọc đang ở Cylinder 11
- Các yêu cầu mới xuất hiện và muốn đọc các cylinder lần lượt là 1, 36, 16, 34, 9 và 12

FCFS (First Come First Served)

- Các yêu cầu phục vụ của các cylinder được ghi vào hàng đợi. Cylinder nào có yêu cầu phục vụ trước thì đầu từ đọc/ghi sẽ dịch chuyển tới đó trước.
- Để đọc được hết các cylinder lần lượt là 1, 36, 16, 34, 9 và 12 thì cách tay đĩa phải di chuyển qua 111 cylinder

SSF (Shortest Seek First)

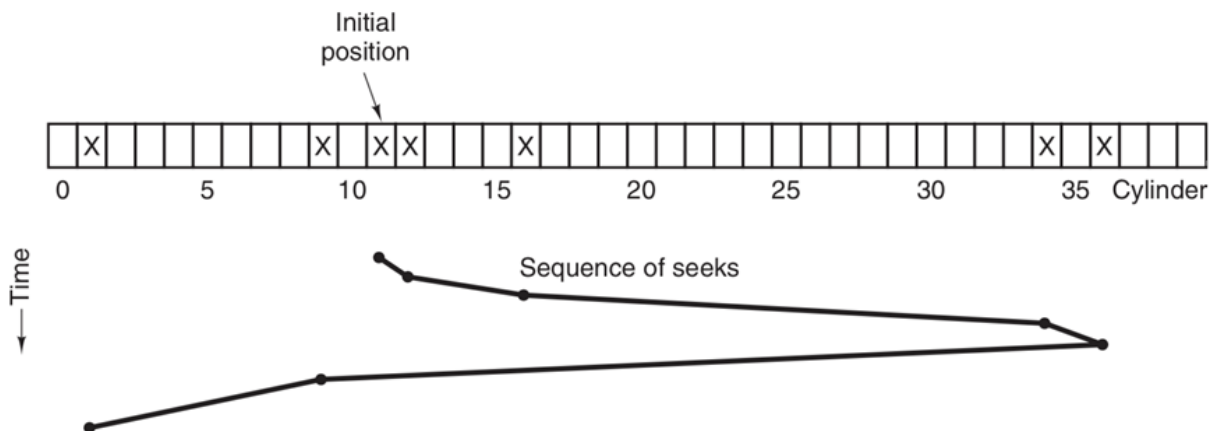
- Cylinder nào có thời gian di chuyển đầu từ đọc/ghi ngắn nhất thì phục vụ trước



- Cách tay đĩa phải chuyển động tổng cộng qua **61** cylinder

Thuật toán thang máy (elevator algorithm)

- Sử dụng một bit cho hướng hiện tại là UP hoặc DOWN. Thang sẽ di chuyển theo cùng hướng cho tới khi không còn request nào theo hướng đó thì mới đổi hướng.
- Giả sử bit định hướng ban đầu có giá trị là UP, Cách tay đĩa phải chuyển động tổng cộng qua 60 cylinder



Bài tập

- Các yêu cầu tới ổ đĩa lần lượt cho các cylinders 10, 22, 20, 2, 40, 6 và 38. Thời gian tìm kiếm là 6ms/cylinder. Giả sử cánh tay đang ở cylinder 20. Tính thời gian tìm kiếm cần thiết khi sử dụng các thuật toán:
 - a) FCFS
 - b) SSF
 - c) Elevator algorithm (khởi tạo bit chỉ hướng UP)
 - d) Elevator algorithm (khởi tạo bit chỉ hướng DOWN)



Hết Chương 5