MÔN HỆ ĐIỀU HÀNH

Chương 6

QUẢN LÝ THIẾT BỊ I/O (NHẬP/XUẤT)

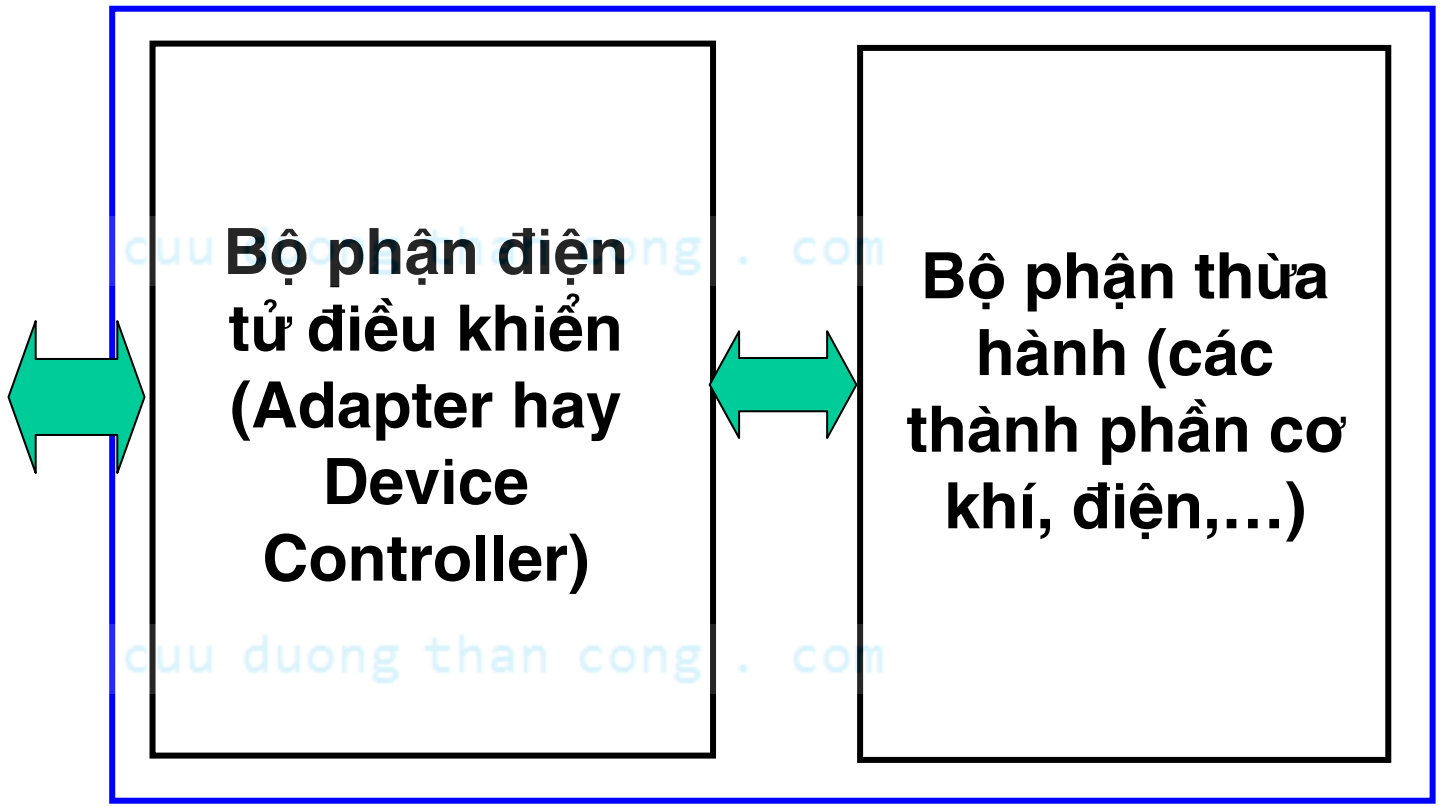
6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O
6.2 Các nguyên tắc cơ bản về phần mềm thiết bị I/O
6.3 Các cấp chức năng cơ bản của hệ thống phần mềm I/O
6.4 Đĩa cứng
6.5 Mạch đồng hồ
6.6 Terminal giao tiếp trên cơ sở từng ký tự
6.7 Giao tiếp người dùng trên cơ sở đồ họa
6.8 Terminal mạng
6.9 Thiết bị quản lý việc dùng năng lượng

Tài liệu tham khảo : chương 5, sách "Modern Operating Systems", Andrew S. Tanenbaum: , 2nd ed, Prentice Hall

6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O

- Thiết bị I/O của máy tính rất đa dạng về chủng loại và chức năng, mặc dù vậy, chúng thường được xây dựng theo nguyên tắc chung như sau :

Connector giao tiếp theo 1 chuẩn xác định (SCSI, SATA, IDE, USB, COM, …) để phục vụ một chuẩn nghi thức giao tiếp xác định (protocol)



6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O

- Về cách thức giao tiếp dữ liệu với thế giới bên ngoài, các thiết bị I/O thường thuộc 1 trong 2 loại : thiết bị block và thiết bị ký tự.
  - Thiết bị block : giao tiếp với bên ngoài từng lần 1 block dữ liệu với độ dài cố định, mỗi block có địa chỉ cố định và độc lập, ta chỉ cần biết địa chỉ của khối thông tin là có thể truy xuất được nó bất kỳ lúc nào.
  - Thiết bị ký tự : giao tiếp với bên ngoài từng lần 1 chuỗi byte có độ dài tùy ý (stream), tuy nhiên stream không có địa chỉ, nghĩa là ta chỉ có thể truy xuất dữ liệu theo dạng tuần tự từ đầu đến cuối, không thể quay lại quá khứ được. Đa số các thiết bị I/O mà ta dùng với máy tính đều thuộc loại thiết bị ký tự (bàn phím, chuột, card mạng, scanner, printer,…)

6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O

- Cũng có 1 vài ngoại lệ :
  - Thiết bị clock : nó không có dữ liệu, chỉ kích hoạt tín hiệu ngắt quảng theo chu kỳ xác định trước.
  - Màn hình có nội dung hiển thị nằm trong RAM. Máy tính chỉ cần dùng các lệnh máy truy xuất vùng RAM tương ứng, màn hình sẽ hiển thị ngay lập tức kết quả bị hiệu chỉnh.

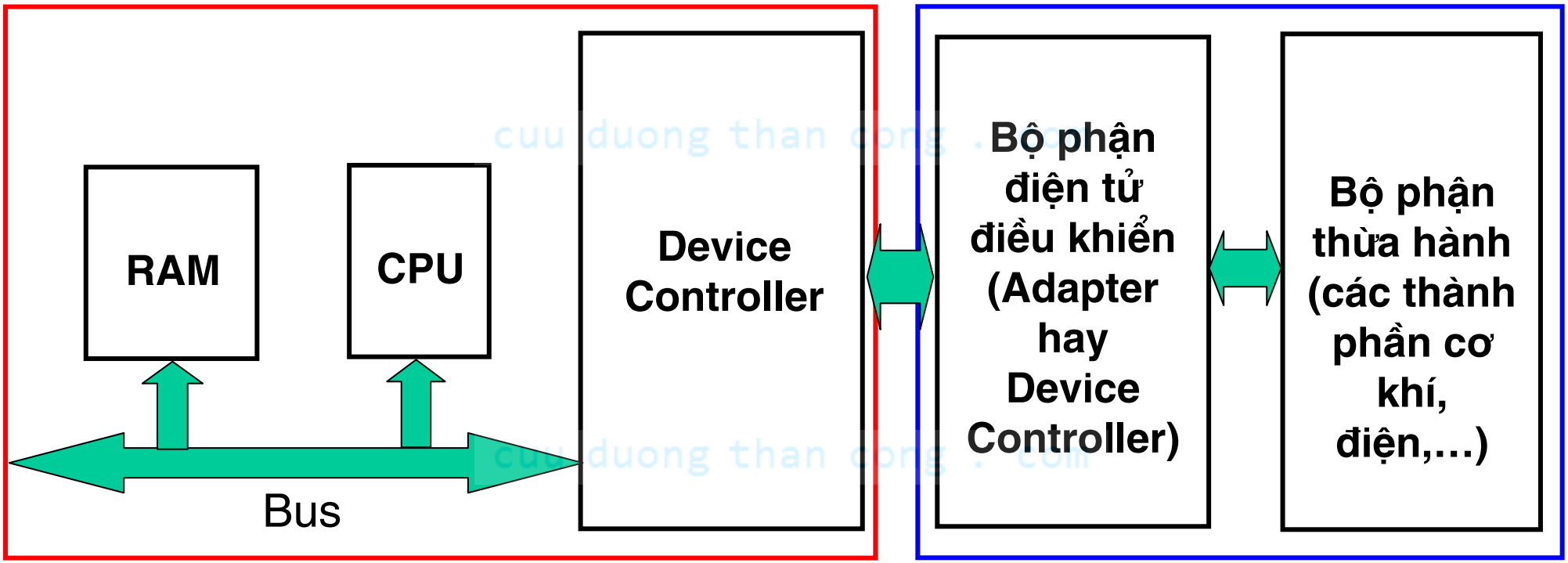
6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O

Các thiết bị I/O có tốc độ giao tiếp rất khác nhau tùy tính chất sử dụng của chúng.

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

6.1 Các nguyên tắc cơ bản về phần cứng thiết bị I/O

- Thiết bị I/O có thể giao tiếp với bất kỳ thiết bị nào khác. Để máy tính có thể giao tiếp với thiết bị I/O, người ta thường dùng 1 bộ phận tương thích với bộ phận điều khiển của thiết bị, ta gọi bộ phận này bên máy tính là device controller :



Các thanh ghi I/O

Mỗi device controller đều có những ô nhớ chứa các thông tin hoạt động, ta gọi các ô nhớ này là các thanh ghi (register). Dựa vào nội dung mà thanh ghi chứa, có 4 loại thanh ghi :

- Thanh ghi lệnh : thanh ghi chứa mã lệnh chức năng mà CPU ghi vào để bắt controller thực hiện. Chiều di chuyển thông tin của thanh ghi này là từ CPU đến controller (OUT).
- Thanh ghi trạng thái : thanh ghi chứa các bit thông tin miêu tả trạng thái hiện hành của thiết bị I/O tương ứng (bận, rãnh,…). Chiều di chuyển thông tin của thanh ghi này là từ controller về CPU (IN).
- Thanh ghi dữ liệu xuất : chứa dữ liệu mà CPU muốn xuất ra thiết bị I/O. Chiều di chuyển thông tin của thanh ghi này là từ CPU đến controller (OUT).
- Thanh ghi dữ liệu nhập : chứa dữ liệu mà thiết bị I/O gởi về máy tính. Chiều di chuyển thông tin của thanh ghi này là từ controller về CPU (IN).

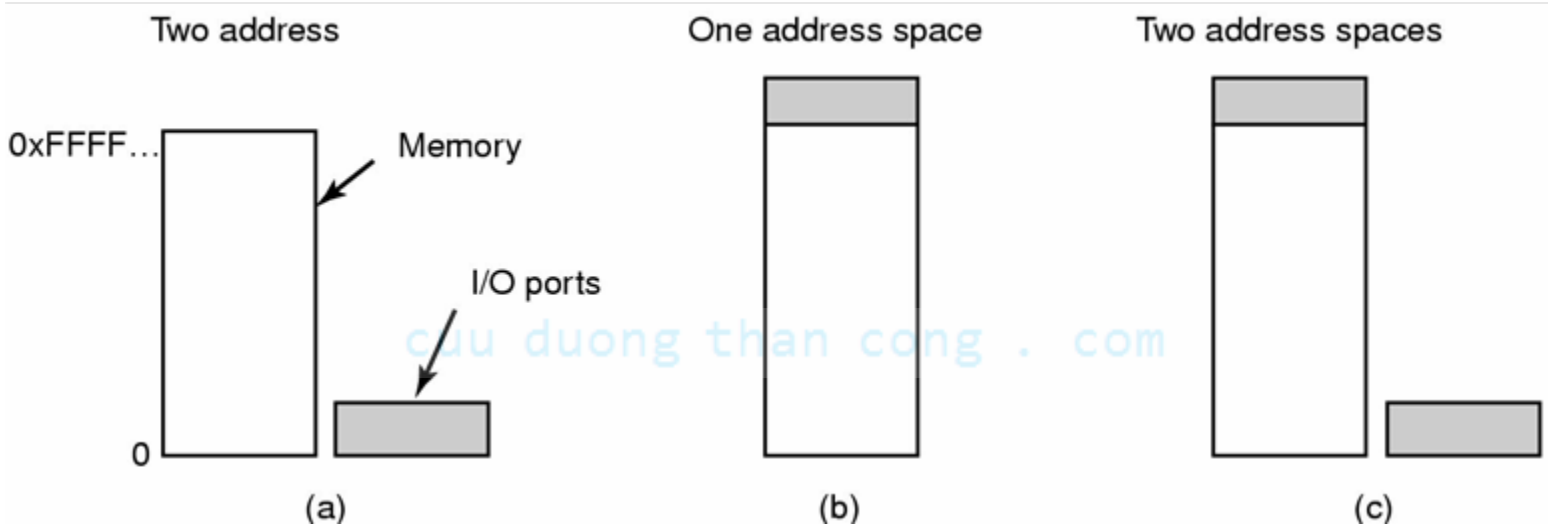
Các thanh ghi I/O

Mỗi thanh ghi cần có địa chỉ truy xuất duy nhất. Có 3 phương pháp gán địa chỉ cho các thanh ghi của 1 controller :

- Địa chỉ I/O : máy có 2 loại địa chỉ khác nhau : địa chỉ ô nhớ dành để truy xuất các ô nhớ trong RAM, địa chỉ I/O dành truy xuất các thanh ghi của các mạch controller. Thí dụ lệnh mov al, [f5] sẽ đọc nội dung ô nhớ RAM ở địa chỉ $F5_H$ vào thanh ghi al của CPU, còn lệnh in al, f5 sẽ đọc nội dung thanh ghi của controller nào đó mà có địa chỉ (port) là $F5_H$.
- Địa chỉ memory-mapped I/O : máy chỉ có 1 loại địa chỉ và 1 loại lệnh để truy xuất các ô nhớ, mỗi thanh ghi I/O phải được thiết kế sao cho nó chiếm 1 địa chỉ ô nhớ xác định, muốn truy xuất thanh ghi I/O, CPU sẽ dùng lệnh truy xuất ô nhớ như bình thường. Thí dụ lệnh mov al, [f5] sẽ đọc nội dung ô nhớ RAM hay thanh ghi I/O ở địa chỉ $F5_H$ tùy thuộc vào địa chỉ này đang được dùng cho phần tử nào.

Các thanh ghi I/O

- Dùng hổn hợp 2 loại địa chỉ I/O và địa chỉ memory-mapped I/O : máy có 2 loại địa chỉ khác nhau : địa chỉ ô nhớ dành để truy xuất các ô nhớ trong RAM, địa chỉ I/O dành truy xuất các thanh ghi của các mạch controller. Tùy theo tính chất sử dụng của từng thanh ghi, ta sẽ dùng địa chỉ I/O (port) hay địa chỉ memory để truy xuất nó rồi dùng loại lệnh tương ứng để truy xuất nó khi cần thiết.

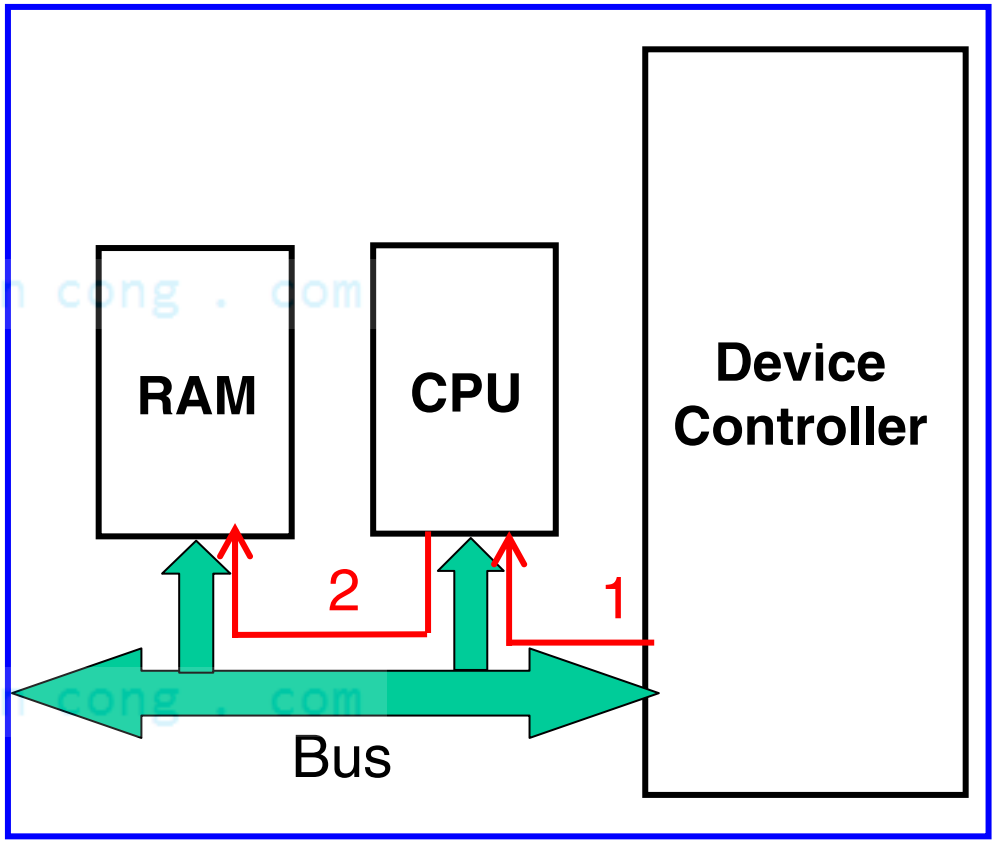


Kỹ thuật DMA

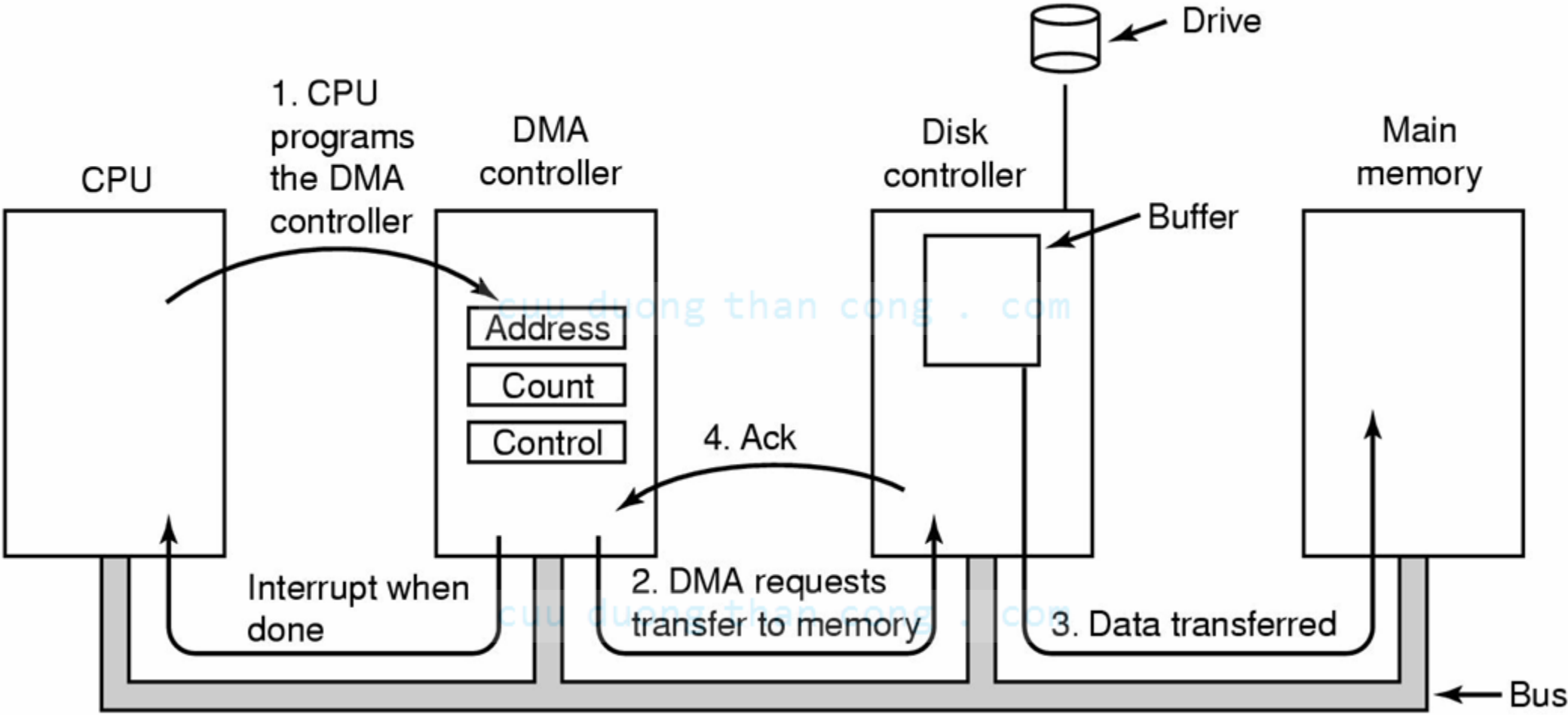
Mỗi lần cần di chuyển dữ liệu từ thiết bị I/O (đang nằm trong thanh ghi data in của controller), CPU phải thực hiện 2 lệnh máy liên tiếp :

1. Lệnh in (hay mov) để di chuyển dữ liệu từ controller vào thanh ghi của CPU.
2. Lệnh mov để di chuyển dữ liệu từ thanh ghi CPU ra ô nhớ RAM xác định.

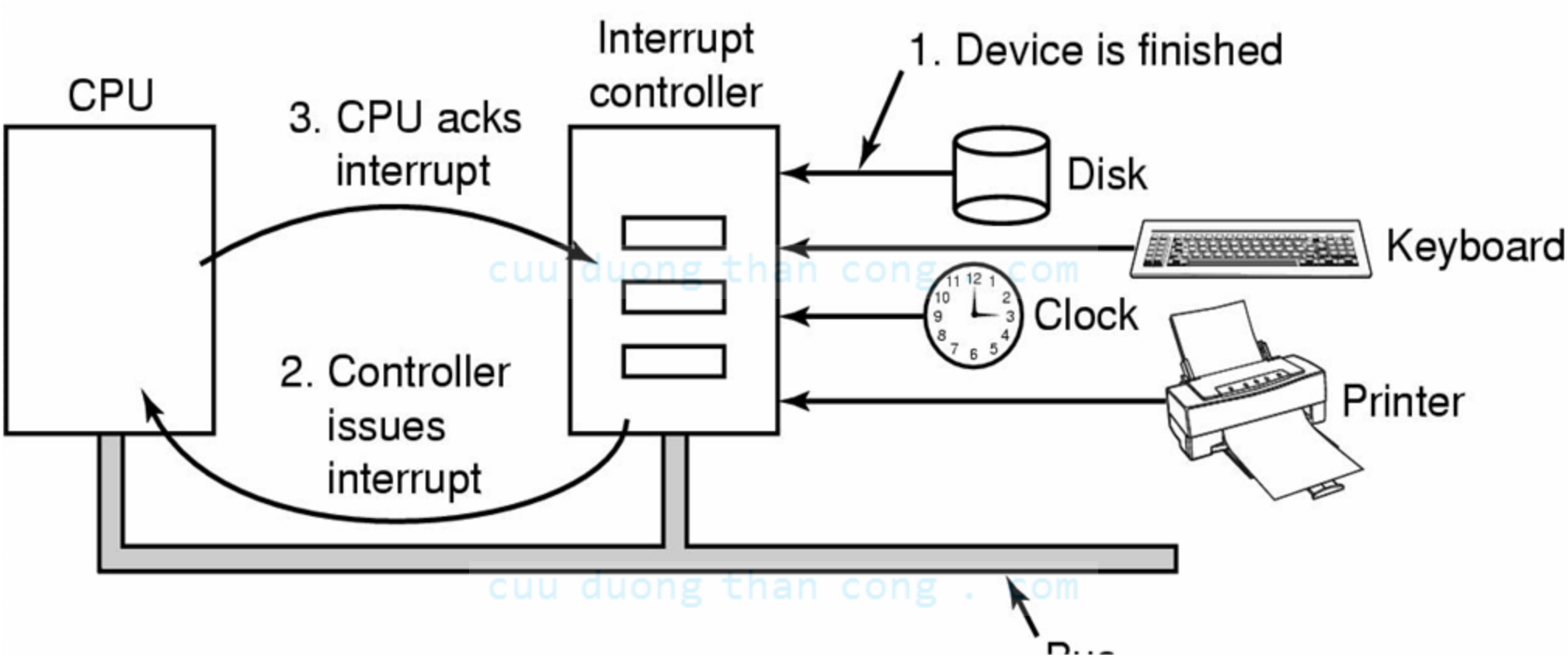
Việc di chuyển dữ liệu giữa máy tính và thiết bị I/O như trên là chưa được hiệu quả vì phải di chuyển qua phần tử trung gian (CPU). Để khắc phục nược điểm này, ta có thể sử dụng kỹ thuật DMA (Direct Memory Access)



Kỹ thuật DMA



Kỹ thuật ngắt quảng (Interrupt)



6.2 Các nguyên tắc cơ bản về phần mềm thiết bị I/O

- Độc lập thiết bị : ứng dụng có thể truy xuất bất kỳ thiết bị nào, qui trình truy xuất phải đồng nhất, không phụ thuộc vào tính chất vật lý hay loại thiết bị.

  Sort <input >output
- Đặt tên thiết bị đồng nhất : tên thiết bị là 1 chuỗi hay 1 số ngyên giống như tên 1 file.
- Che dấu việc xử lý lỗi : nếu có lỗi trong khi truy xuất I/O, hệ thống phần mềm I/O phải xử lý tốt nhất và ở cấp thấp nhất rồi cố gắng che dấu lỗi càng nhiều càng tốt.
- Chuyển đổi cung cách nhập/xuất dữ liệu dạng bất đồng bộ thành dạng đồng bộ : Ở cấp vật lý, mỗi lần cần đọc 1 sector đĩa, CPU sẽ xuất lệnh đọc ra controller, chờ ngắt khi có dữ liệu, đọc khối dữ liệu vào bộ nhớ. Ở cấp ứng dụng, nên cung cấp 1 hàm chức năng, ứng dụng gọi hàm này, khi hàm return kết quả (lâu hay mau), ứng dụng đã có dữ liệu để xử lý tiếp.

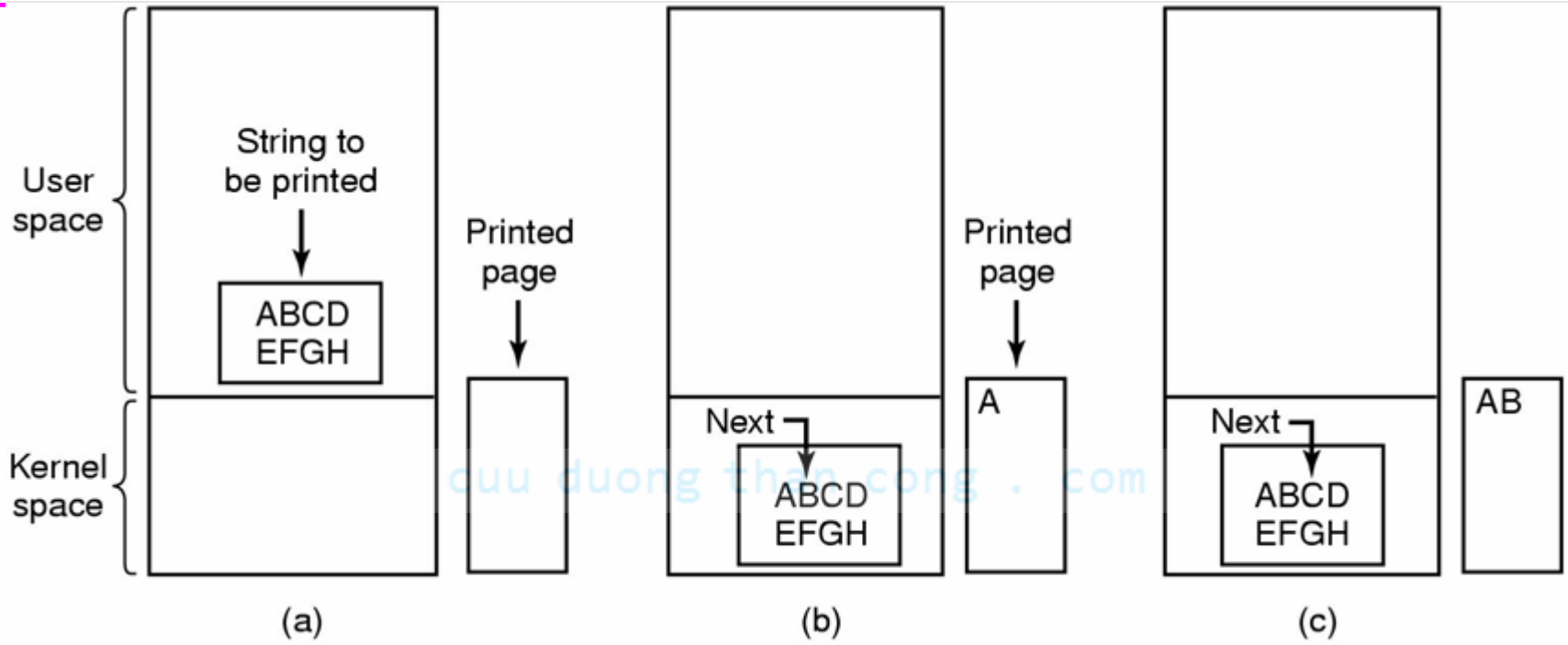
6.2 Các nguyên tắc cơ bản về phần mềm thiết bị I/O

- Đệm dữ liệu nhấp/xuất : khi dữ liệu từ 1 thiết bị I/O vào máy tính, HĐH chưa biết dữ liệu này sẽ được ứng dụng nào dùng, mà biết ứng dụng nào đi nữa thì cũng khó lòng đòi hỏi phần mềm đó xử lý kịpthời dữ liệu nhập. Do đó hệ thống phần mềm I/O cần đệm dữ liệu ở 1 nơi nào đó rồi cung cấp cho ứng dụng khi có yêu cầu.
- Thiết bị dùng chung (shared) hay thiết bị dùng riêng (delicated) : thí dụ disk có thể phục vụ nhiều ứng dụng đồng thời, còn băng từ thì chỉ phục vụ 1 ứng dụng tại 1 thời điểm, nếu không thời gian đáp ứng sẽ bị trì hoản rất lâu, nhiều khi là vô tận.

3 cách thực hiện I/O khác nhau

1. Lập trình thực hiện I/O (programmed I/O) : thí dụ sau khi nhận chuỗi ký tự cần in, HĐH sẽ lặp kiểm tra trạng thái máy in, nếu rãnh thì xuất ra 1 ký tự cho đến hết chuỗi.
2. Dùng cơ chế ngắt (interrupt) : thí dụ sau khi nhận chuỗi ký tự cần in, HĐH sẽ khởi động chế độ phục vụ ngắt từ máy in rồi làm việc khác. Mỗi khi máy in rãnh, nó tạo tín hiệu yêu cầu ngắt gởi tới CPU, máy sẽ thực hiện thủ tục đáp ứng ngắt, thủ tục này sẽ kiểm tra xem còn ký tự cần in không ? Nếu hết thì thôi, nều còn thì xuất ký tự kế tiếp ra máy in rồi quay lại nơi đang xử lý để tiếp tục công việc.
3. Dùng kỹ thuật DMA (DMA) : thí dụ sau khi nhận chuỗi ký tự cần in, HĐH sẽ khởi tạo 1 session DMA cho DMA controller biết. DMA controller sẽ thực hiện xuất từng ký tự ra máy in khi cần, không có đoạn code nào của HĐH hay của ứng dụng liên quan đến việc xuất ký tự cụ thể cả.

Lập trình thực hiện I/O (programmed I/O)




```
copy_from_user(buffer, p, count);          /* p is the kernel bufer */
for (i = 0; i < count; i++) {              /* loop on every character */
      while (*printer_status_reg != READY) ;   /* loop until ready */
      *printer_data_register = p[i];           /* output one character */
}
return_to_user( );
```

Dùng cơ chế ngắt (interrupt)

```
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)

Dùng kỹ thuật DMA (DMA)

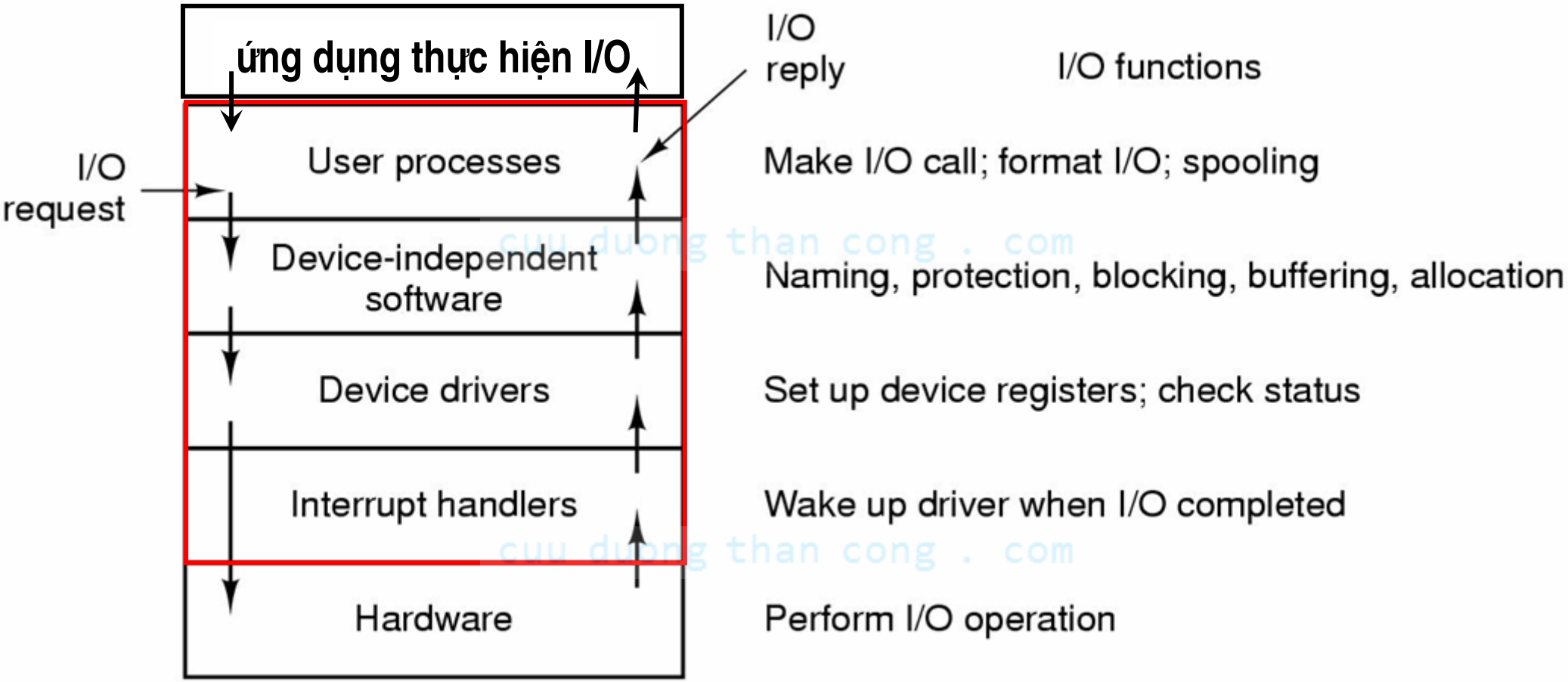
```
copy_from_user(buffer, p, count);
set_up_DMA_controller( );
scheduler( );
```

(a)

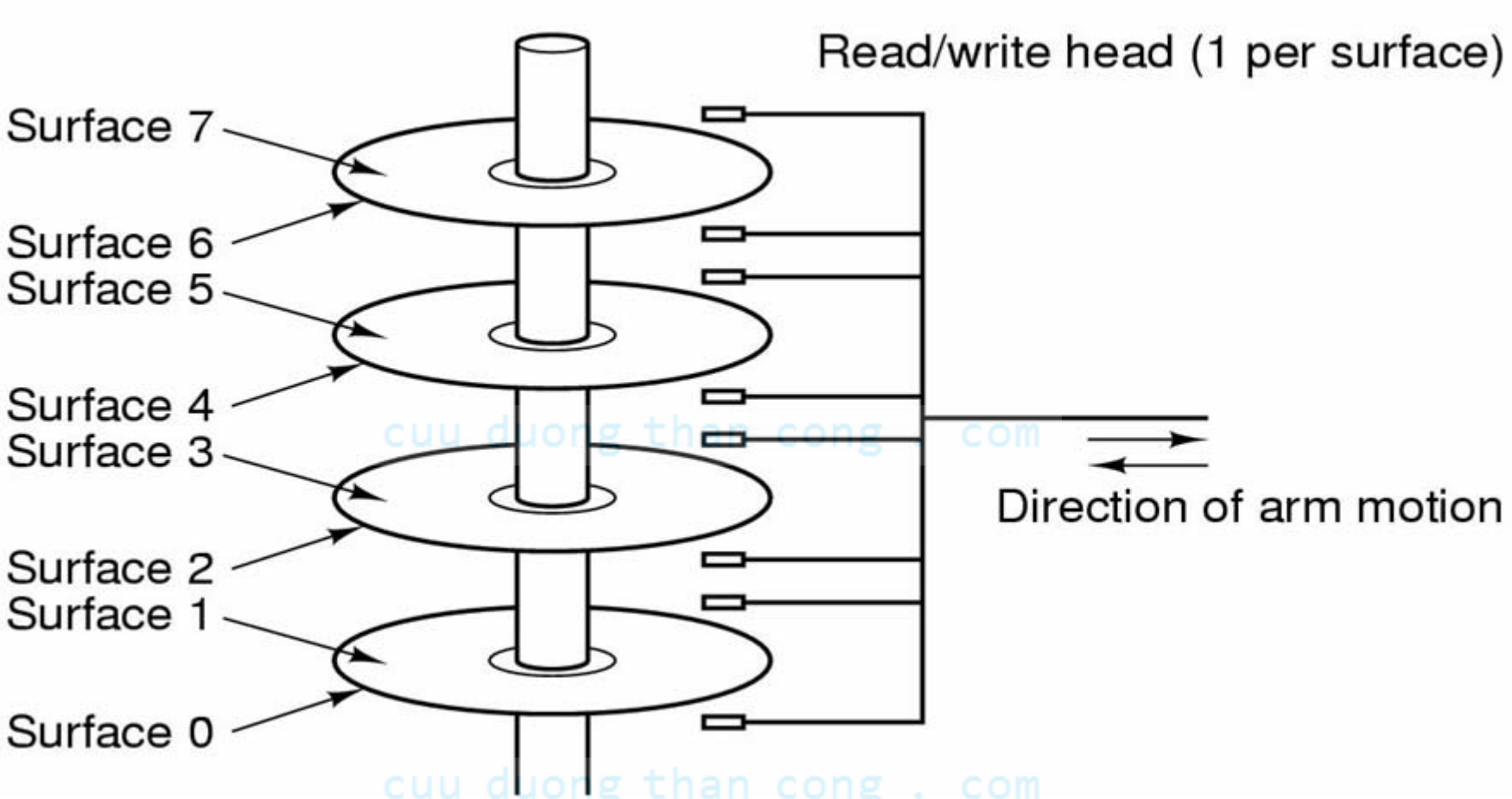
```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

6.3 Các cấp chức năng cơ bản của hệ thống phần mềm I/O



6.4 Đĩa cứng & CDROM

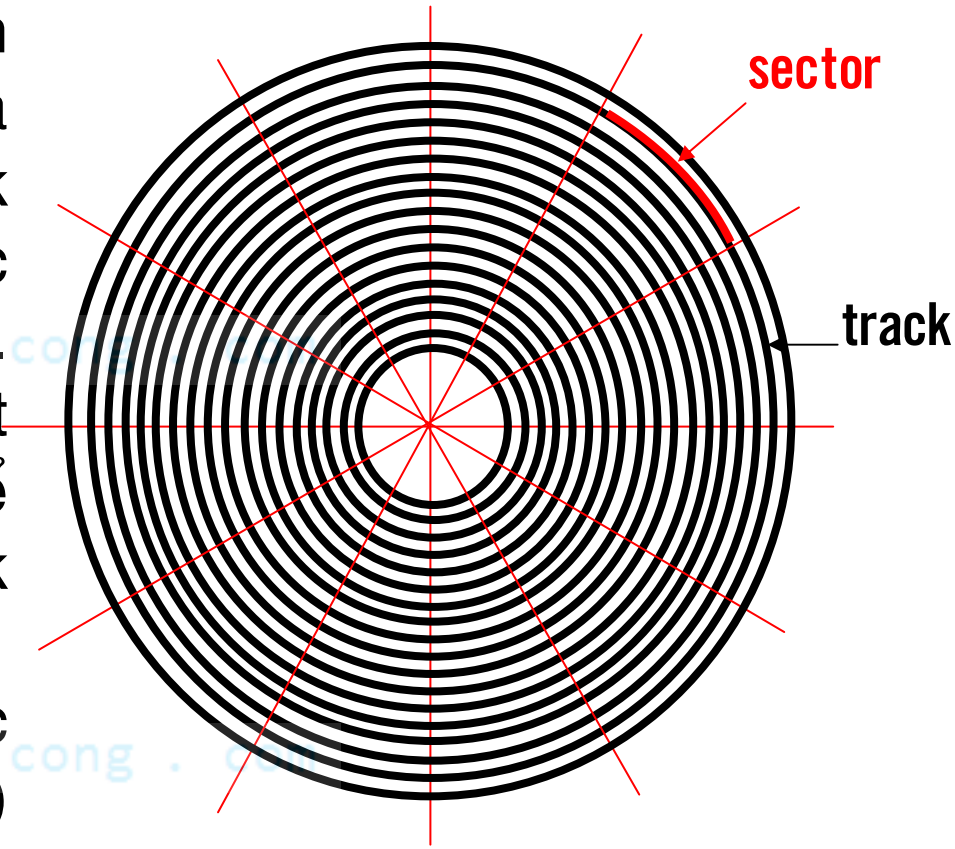


Cấu trúc của một ổ đĩa cứng

6.4 Đĩa cứng & CDROM

Theo góc nhìn từ ngoài, disk vật lý là không gian dữ liệu 3 chiều, mỗi disk = nhiều cylinder, mỗi cylinder gồm nhiều track (head – vòng tròn chứa tin - có cùng đường kính), mỗi track chứa nhiều cung chứa tin nhỏ được truy xuất độc lập nhau (sector). Sector là đơn vị truy xuất tin nhỏ nhất ở cấp vật lý (từ ngoài ta không thể truy xuất từng byte dữ liệu trên disk được).

Muốn truy xuất 1 sector, ta phải xác định tọa độ 3 chiều của nó (C,H,S) → rất khó tư duy.



Đĩa cứng RAID 0

Ta có thể xem disk hiện thực y như góc nhìn truy xuất từ bên ngoài là SLED (Single Large Expensive Disk). Đĩa SLED thường không có tốc độ truy xuất cao và độ tin cậy thấp. Để khắc phục 2 nhược điểm trên, người ta thường dùng kỹ thuật RAID (Redudant Array of Inpependant Disks). Có 6 dạng RAID khác nhau như sau :

RAID 0 : gộp k sector thành 1 strip, dùng n disk thô để hiện thực disk luận lý như sau : phân bổ các strip theo kiểu round-robin như hình dưới đây. Nếu máy tính truy xuất khối dữ liệu dài n strip, controller trên RAID0 sẽ biết được khối dữ liệu n strip này nằm trên n disk thô khác nhau, nó sẽ ra lệnh n disk đồng thời, mỗi disk thô truy xuất 1 trip. Dữ liệu truy xuất được sẽ được hợp lại/phân ra bởi controller.

Đĩa cứng RAID 1

RAID 0 đã tăng tốc độ truy xuất lên n lần, nhưng độ tin cậy sẽ bị giảm đi n lần. Để khắc phục nhược điểm này, ta có thể dùng RAID 1.

RAID 1 : dùng cơ chế Master/Miror. Ngoài n disk thô có sẵn (master), dùng thêm n disk thô nữa (slave). Mỗi lần cần ghi thông tin, controller sẽ double dữ liệu và yêu cầu ghi đồng thời lên cả master lẫn slave. Mỗi khi cần đọc thông tin, controller chỉ cần yêu cầu master thực hiện. Chỉ trong trường hợp master có trục trặc, controller mới switch vai trò của master và slave cho nhau rồi thực hiện lại việc đọc dữ liệu (bây giờ trên các disk thô tốt nên sẽ thành công). Người quản trị chỉ cần thay thế bộ phận bị lỗi và copy dữ liệu từ bộ phận còn tốt sang là hệ thống sẽ trở lại trạng thái bình thường như xưa.

Đĩa cứng RAID 2

RAID 1 sử dụng quá nhiều disk thô. Để khắc phục nhược điểm này, ta có thể dùng RAID 2.

RAID 2 : để lưu 32 bit dữ liệu, controller sẽ đổi 32 bit data thành mã sửa sai Hamming 39 bit, từng bit sẽ được ghi lên 1 disk thô khác nhau đồng thời. Như vậy về mặt tốc độ, ta nâng lên được 32 lần so với đĩa SLED. Về độ tin cậy, nếu có lỗi ở 1 disk thô nào đó, nhờ mã Hamming, controller sẽ tự sửa sai để xác định 32 bit dữ liệu gốc. Chi phí giảm từ 2/1 ở RAID 1 xuống còn 39/32 ở RAID 2.

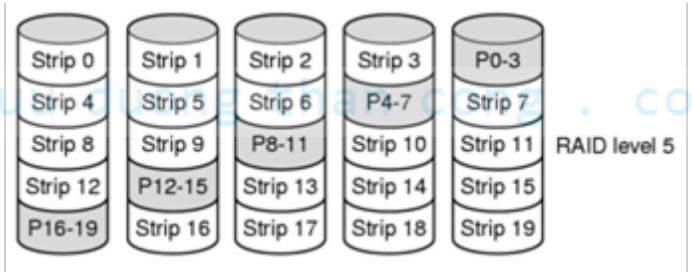
Đĩa cứng RAID 3

RAID 3 : ý tưởng gần giống với RAID 2 : để lưu 32 bit dữ liệu, controller sẽ dùng 33 disk thô : 32 disk chứa 32 bit data, disk còn lại chứa bit phát hiện lỗi (parity). Như vậy về mặt tốc độ, ta nâng lên được 32 lần so với đĩa SLED. Về độ tin cậy, nếu có lỗi ở 1 disk thô nào đó, nhờ bit parity ta phát hiện được lỗi, rồi nhờ tín hiệu báo lỗi ở disk thứ i, controller sẽ tự sửa sai bit i để xác định 32 bit dữ liệu gốc. Chi phí giảm từ 2/1 ở RAID 1 xuống còn 39/32 ở RAID 2 và còn 33/32 ở RAID 3.

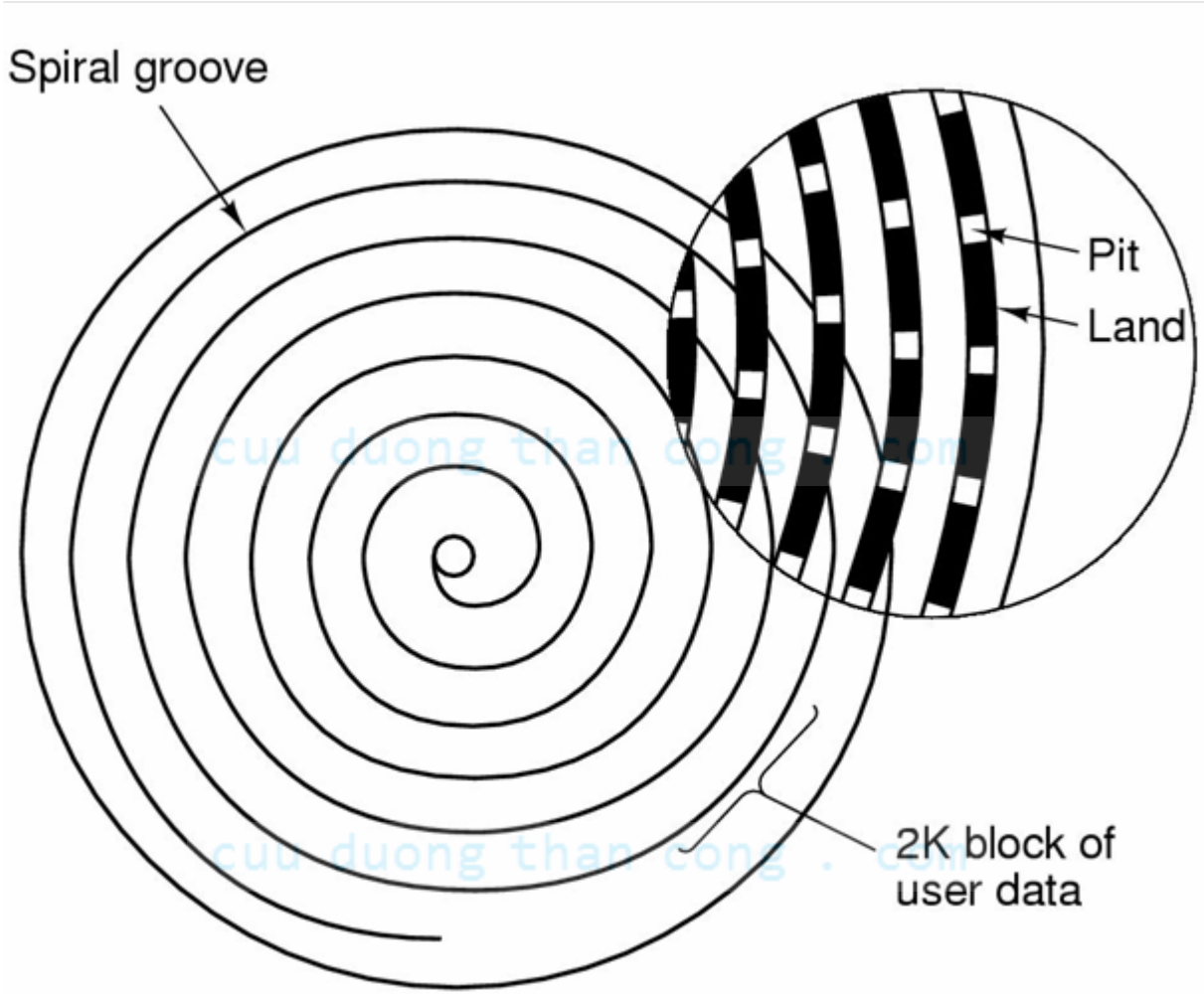
Đĩa cứng RAID 4

RAID 4 : giống như RAID 0, nhưng tăng cường thâm 1 disk thơ để chứa strip partity của n strip nằm trân n disk thĩ chứa dữ liệu. : để lưu 32 bit dữ liệu, controller sẽ dùng 33 disk thô : 32 disk chứa 32 bit data, disk còn lại chứa bit phát hiện lỗi (parity). Như vậy về mặt tốc độ, ta nâng lên được 32 lần so với đĩa SLED. Về độ tin cậy, nếu có lỗi ở 1 disk thô nào đó, nhờ bit parity ta phát hiện được lỗi, rồi nhờ tín hiệu báo lỗi ở disk thứ i, controller sẽ tự sửa sai bit i để xác định 32 bit dữ liệu gốc. Chi phí giảm từ 2/1 ở RAID 1 xuống còn 39/32 ở RAID 2 và còn 33/32 ở RAID 3.

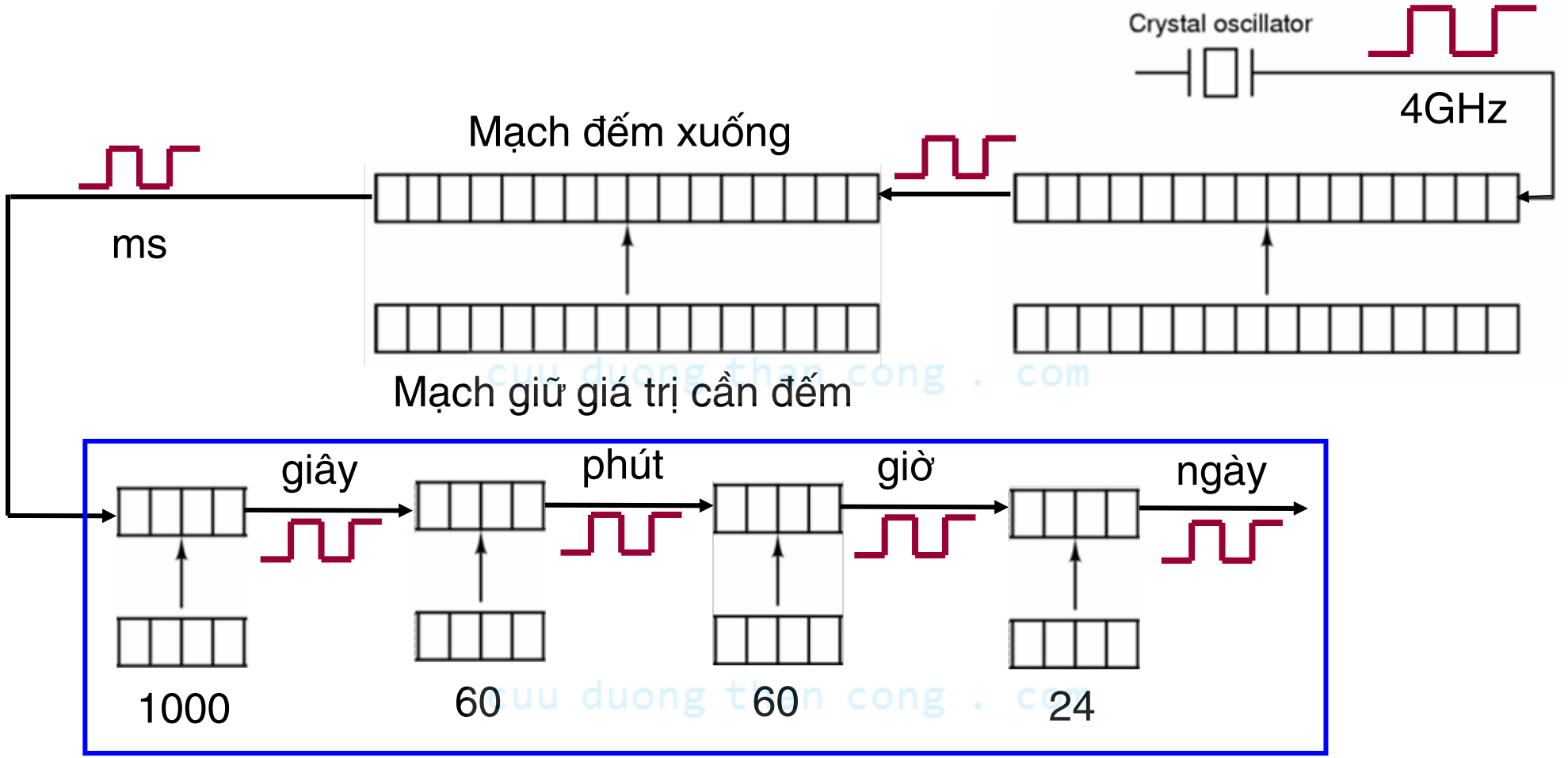
Đĩa cứng RAID 5



Kỹ thuật ghi tin trên CDROM



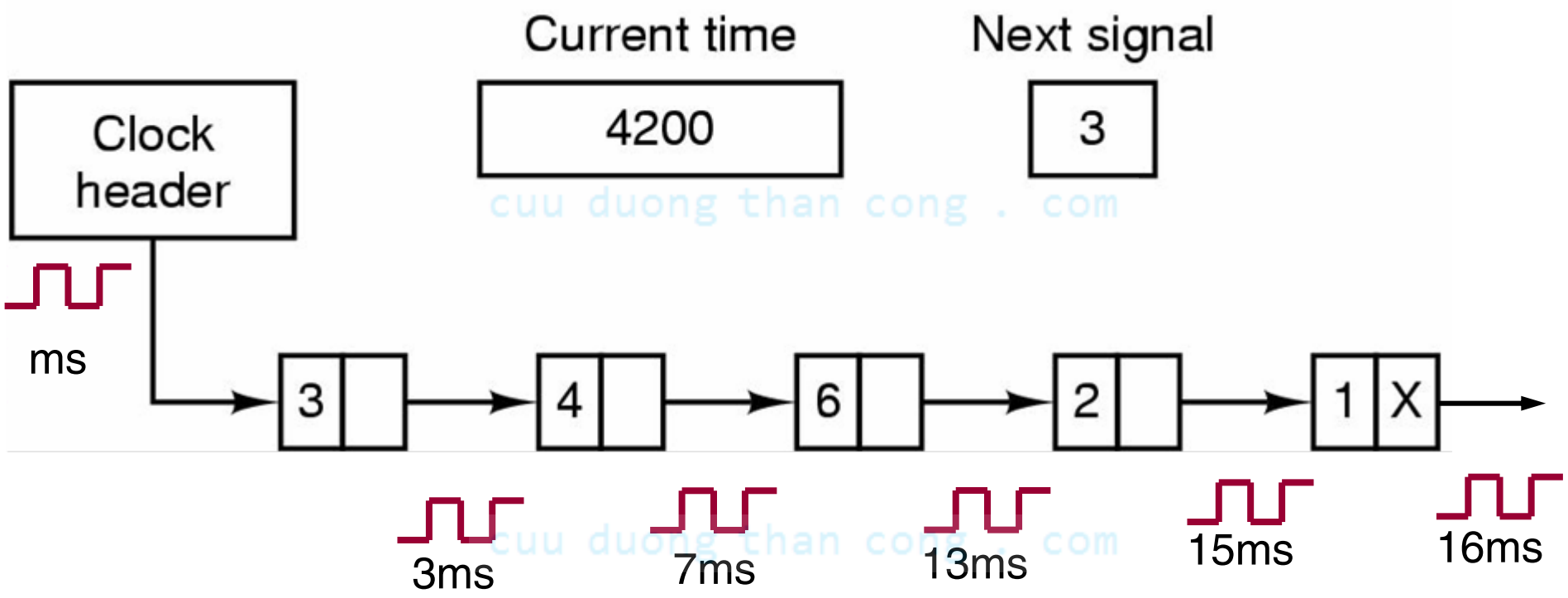
6.5 Mạch đồng hồ



Phần trong hình chữ nhật xanh thường được lập trình để giảm nhẹ phần cứng

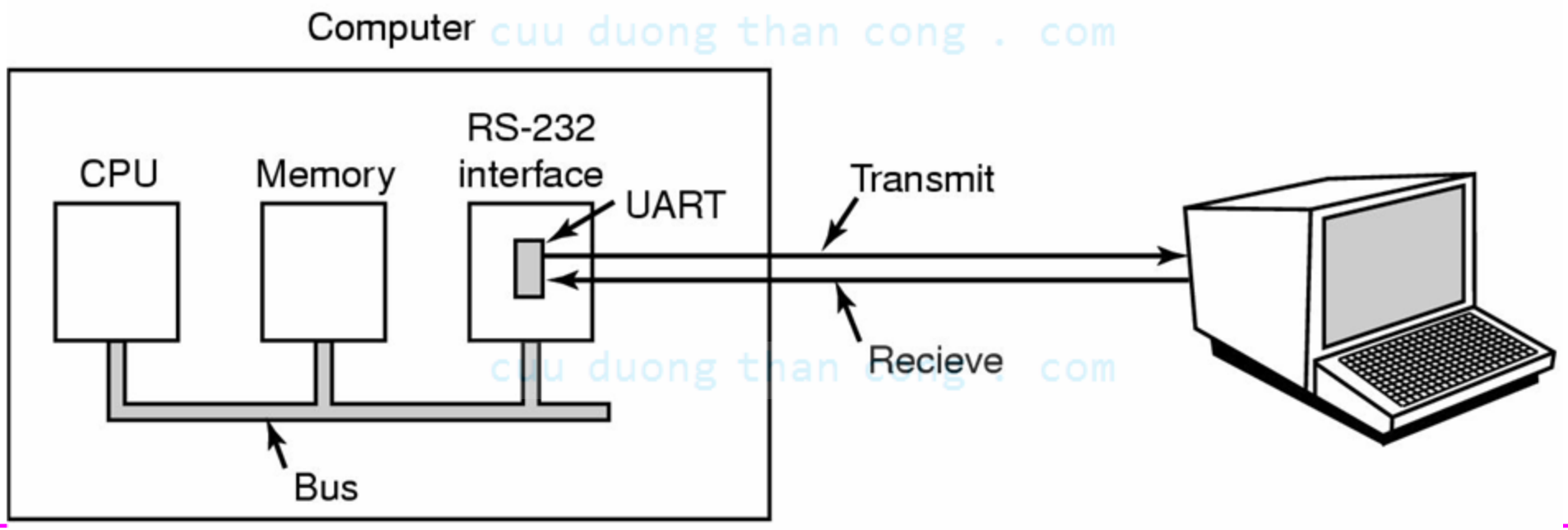
6.5 Mạch đồng hồ

Dùng phần mềm để hiện thực n mạch đếm giờ độc lập



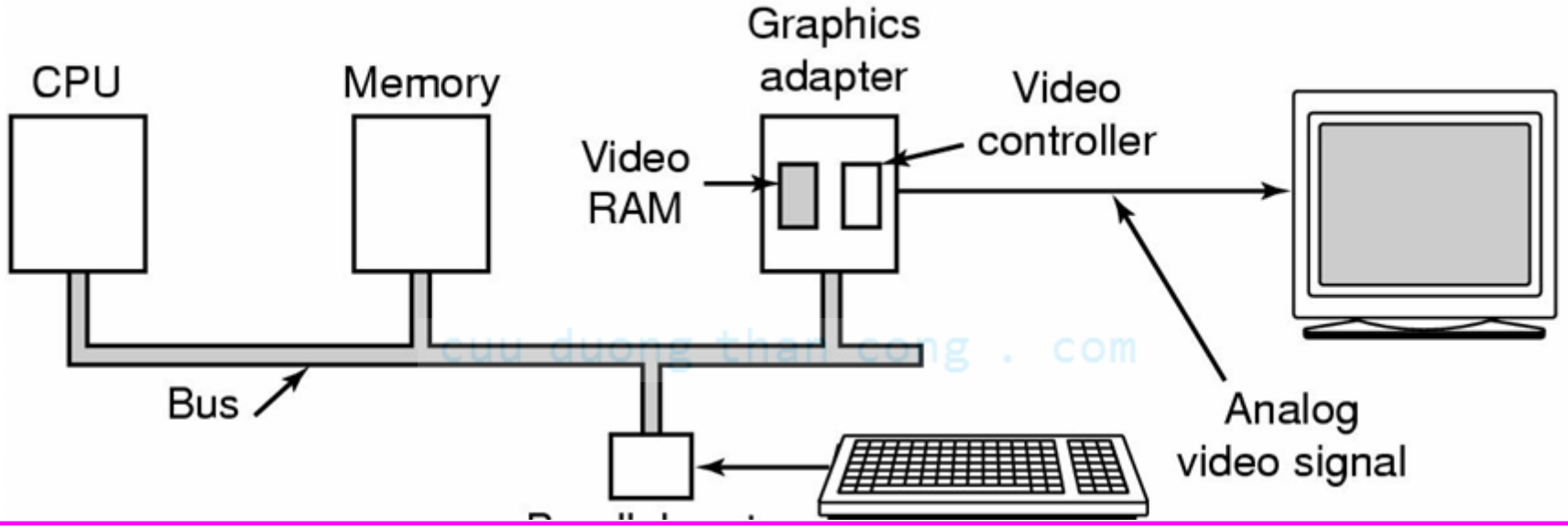
6.6 Terminal giao tiếp trên cơ sở từng ký tự

Terminal và mạch điều khiển bên máy tính dùng chuẩn giao tiếp RS-232 (COM1, COM2) để giao tiếp nhau, tín hiệu được truyền/nhận từng bit theo theo gian. Mỗi lần 1 ký tự ASCII (1byte) cần truyền, controller sẽ đổi ký tự thành 8 bit rồi gởi từng bit ra terminal. Terminal và máy tính là 2 thiết bị độc lập. Terminal thường có khả năng hiển thị được 25 hàng, mỗi hàng 80 ký tự.

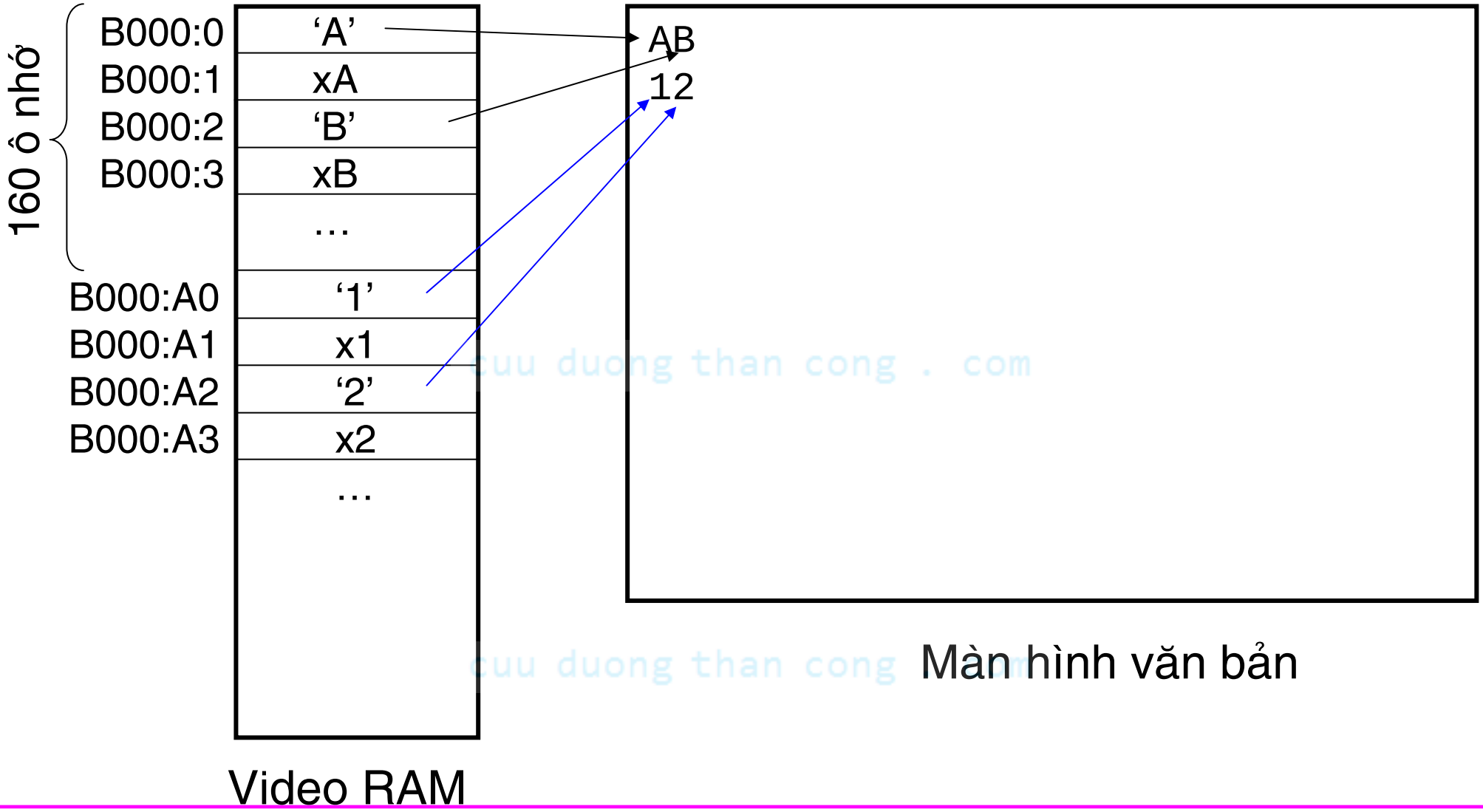


6.7 Giao tiếp người dùng trên cơ sở đồ họa

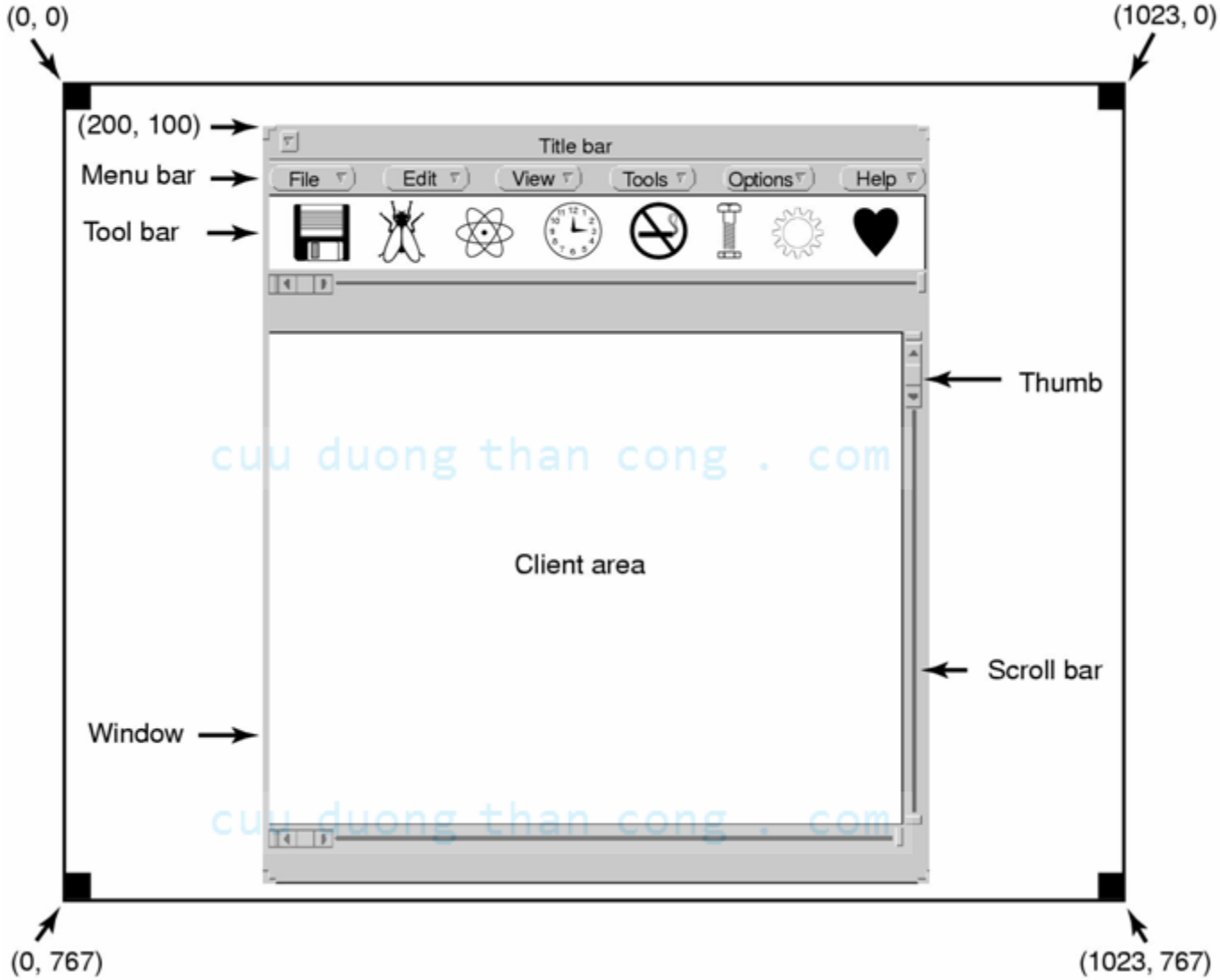
Màn hình đồ họa có khả năng hiển thị c*r pixel cơ bản (1024*768). Thông tin điều khiển hiển thị 1 pixel dài n bit (24bit = 3 byte cho chế độ TrueColor). Ma trận thông tin hiển thị cho c*r pixel được lưu trong bộ nhớ "Video RAM", CPU có thể truy xuất trực tiếp từng ô nhớ trong video RAM y như truy xuất ô nhớ trong RAM chứa chương trình và dữ liệu ⇒ việc cập nhật thông tin hiển thị đồ họa rất nhanh chóng, có thể chiếu phim full HD.



Màn hình ở chế độ văn bản



Màn hình ở chế độ đồ họa

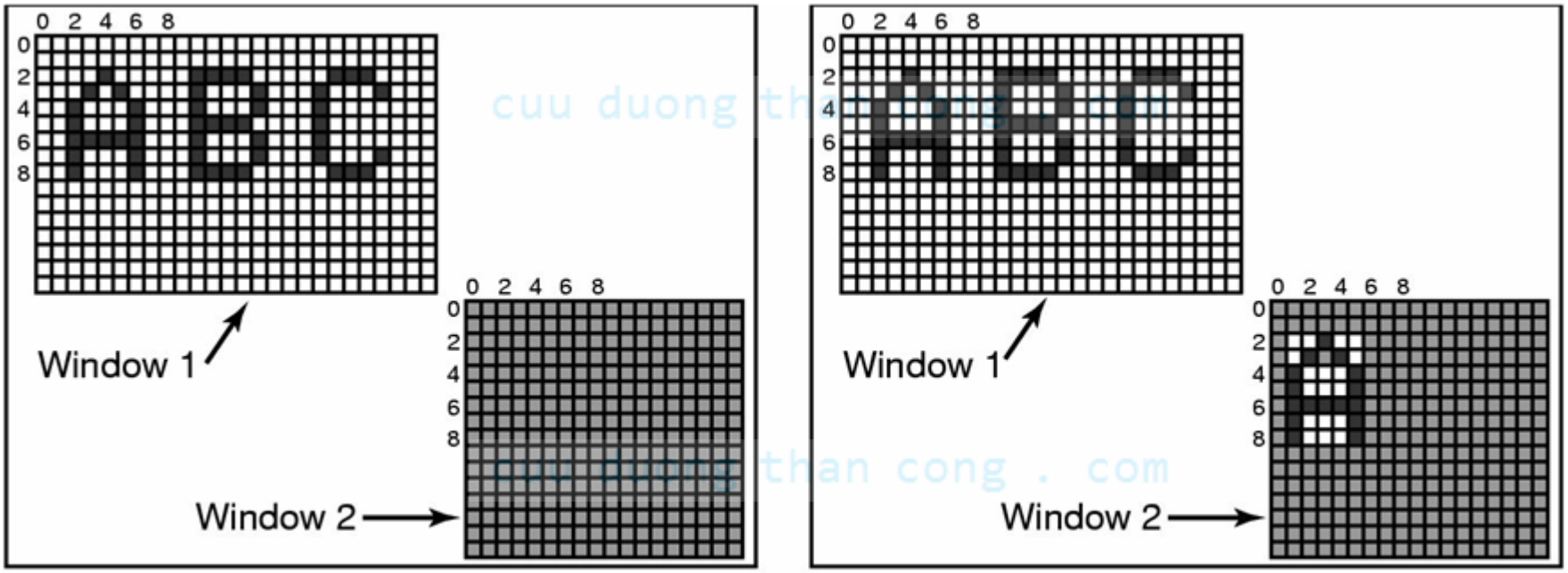


Màn hình ở chế độ đồ họa

Hàm API Windows BitBlt() cho phép copy 1 vùng bitmap gốc sang vùng vẽ :

a. Trước khi hàm BitBlt thực thi.

b. Sau khi hàm BitBit thực thi.

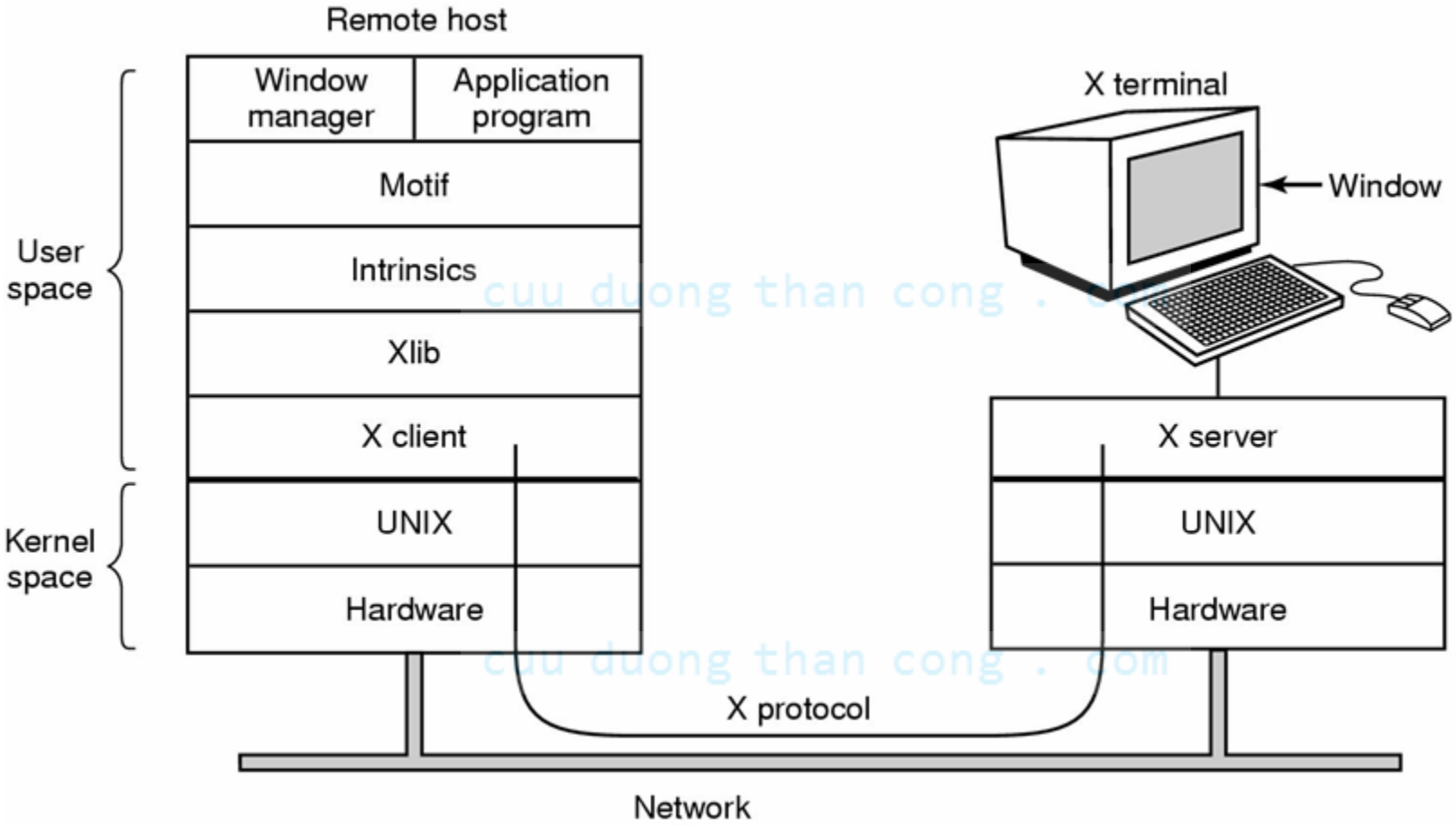


Màn hình ở chế độ đồ họa

Thường dùng font outline để miêu tả đường viền từng ký tự được hiển thị trong chế độ đồ họa.

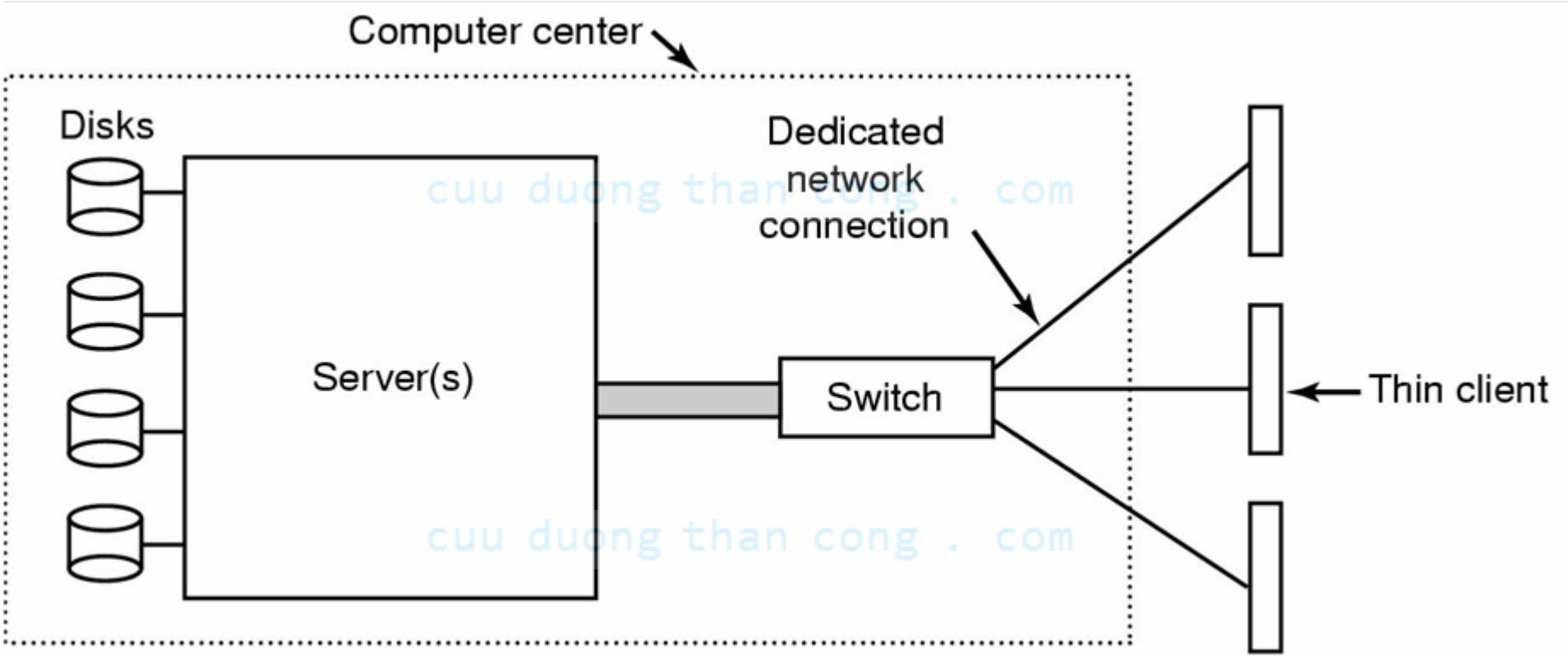
6.8 Terminal mạng

X-Terminal là thiết bị đầu cuối có khả năng hiển thị đồ họa với tốc độ rất cao, được nối kết với máy tính thông qua kỹ thuật mạng máy tính.



Các Terminal mạng làm việc với cùng 1 máy tính

Kiến trúc hệ thống các terminal SLIM (Stateless Low-level Interface Machine).



6.9 Thiết bị quản lý việc dùng năng lượng

1. Máy ENIAC gồm 18.000 đèn điện tử, tốn 140.000w điện ⇒ quá tốn điện.
2. Máy PC dùng transistor nên tốn rất ít điện, khoảng 400w, trong đó máy dùng khoảng 85%, còn lãng phí 15% (60w). Tuy nhiên nếu 1 tỉ máy PC đồng thời chạy trên toàn thế giới, con người đã lãng phí 60 tỉ Watt = 60.000 Mega Watts = công suất phát của 60 nhà máy điện nguyên tử có qui mô trung bình ⇒ quá lãng phí điện.
3. Máy laptop và các thiết bị di động dùng pin charge, để hạn chế trọng lượng của máy, ta phải hạn chế kích thước và trọng lượng của pin ⇒ thời gian cung cấp năng lượng của pin rất hạn chế (vài ba giờ hoạt động). Hơn nữa năng lượng của pin được nạp từ năng lượng điện, do đó cần hạn chế việc sử dụng năng lượng của các thành phần hoạt động của máy xuống.

6.9 Thiết bị quản lý việc dùng năng lượng

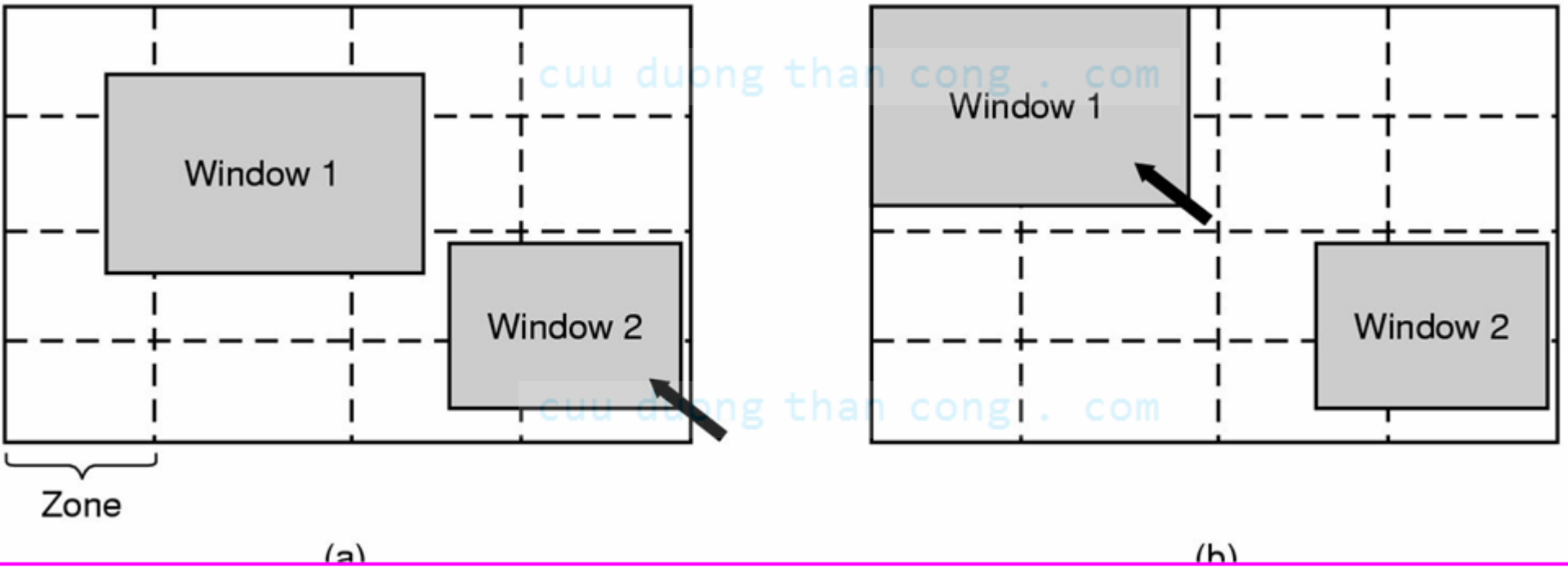
- Ngoài việc chế tạo các thiết bị phần cứng sao cho chúng sử dụng ít điện năng, vai trò của HĐH và ứng dụng cũng rất quan trọng trong việc giảm thiểu việc tiêu tốn điện năng.
- Về phần cứng (CPU, memory, thiết bị I/O), chúng được thiết kế và chế tạo sao cho có thể ở nhiều trạng thái khác nhau như :
  1. On : đang hoạt động ⇒ tốn năng lượng nhất.
  2. Sleeping : ngủ chờ ⇒ ít tốn năng lượng.
  3. Hibernating : ngủ đông ⇒ tốn ít năng lượng hơn nữa, nhưng khi muốn quay về on, nó sẽ tiêu tốn năng lượng và thời gian hơn là từ sleeping về on.
  4. Off : tắt ⇒ không tiêu tốn năng lượng.
- Nhiệm vụ của HĐH và ứng dụng là điều khiển các thiết bị phần cứng để chúng ở trạng thái phù hợp theo thời gian sao cho tổng chi phí tiêu thụ năng lượng là nhỏ nhất.

6.9 Thiết bị quản lý việc dùng năng lượng

- Ngoài việc chế tạo các thiết bị phần cứng sao cho chúng sử dụng ít điện năng, vai trò của HĐH và ứng dụng cũng rất quan trọng trong việc giảm thiểu việc tiêu tốn điện năng.
- Về phần cứng (CPU, memory, thiết bị I/O), chúng được thiết kế và chế tạo sao cho có thể ở nhiều trạng thái khác nhau như :
  1. On : đang hoạt động ⇒ tốn năng lượng nhất.
  2. Sleeping : ngủ chờ ⇒ ít tốn năng lượng.
  3. Hibernating : ngủ đông ⇒ tốn ít năng lượng hơn nữa, nhưng khi muốn quay về on, nó sẽ tiêu tốn năng lượng và thời gian hơn là từ sleeping về on.
  4. Off : tắt ⇒ không tiêu tốn năng lượng.
- Nhiệm vụ của HĐH và ứng dụng là điều khiển các thiết bị phần cứng để chúng ở trạng thái phù hợp theo thời gian sao cho tổng chi phí tiêu thụ năng lượng là nhỏ nhất.

Tiết kiệm năng lượng cho màn hình

- Tắt màn hình sau 1 thời gian nội dung không biến động, người dùng có thể thiết lập khoảng thời gian này sao cho hợp lý.
- Chia màn hình ra nhiều zone độc lập, tại 1 thời điểm nào đó, nếu chỉ có 1 cửa sổ nhỏ đang tác động và làm việc với người dùng, máy có thể tắt các zone khác.



Tiết kiệm năng lượng cho đĩa cứng

- Khi quay, động cơ quay đĩa tiêu tốn rất nhiều điện năng. Do đó nên tắt động cơ quay khi không cần thiết, thí dụ sau 1 khoảng thời gian không truy xuất đĩa, người dùng có thể thiết lập khoảng thời gian này sao cho hợp lý. Lưu ý khi cần truy xuất lại, động cơ phải được khởi động lại và chờ 1 thời gian để tốc độ ổn định, điều này sẽ tiêu tốn điện năng và thời gian chờ nhiều.
- Có thể dùng Cache lớn để hạ thấp tần suất truy xuất đĩa, nhờ đó việc tắt động cơ quay sẽ thuận lợi và có ích hơn nhiều.
- Có thể thông báo cho ứng dụng biết trạng thái "tắt động cơ quay" cho ứng dụng để ứng dụng delay việc truy xuất đĩa nếu có thể.

Tiết kiệm năng lượng cho CPU

- Khi CPU về trạng thái rãnh (hoặc chờ I/O hoặc không có process để chạy), nó nên chuyển về trạng thái sleep.
- Giảm tần số chạy của CPU để giảm mức tiêu thụ điện của nó. Các ứng có deadline biết trước :
  - thí dụ trình xem phim cần đọc/giải mã và hiển thị 25 frame/s, tức thời gian xử lý 1 frame ảnh là 40ms. Nhưng CPU nhanh, nó có thể chỉ cần 20ms để xử lý xong 1 frame. Trong trường hợp này, ta giảm tốc độ chạy CPU xuống 2 lần để giảm mức tiêu hao năng lượng 4 lần.
  - Hoặc nếu người dùng nhập liệu với tốc độ 1ký tự/s, còn ứng dụng chỉ cần 100ms là xử lý xong việc nhập 1 ký tự. Trong trường hợp này, ta giảm tốc độ chạy CPU xuống 10 lần để giảm mức tiêu hao năng lượng 100 lần.

Tiết kiệm năng lượng cho RAM

- Khi không cần thiết, hãy giải phóng nội dung của Cache rồi off nó.
- Khi không cần thiết, hãy lưu nội dung trong RAM lên đĩa rồi off nó (Hibernation).

cuu duong than cong . com