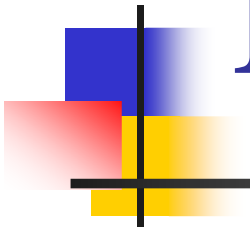
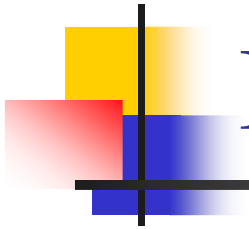


HỆ ĐIỀU HÀNH



Bài 1_Tổng quan

- Điều khiển thực thi các chương trình ứng dụng
- Đóng vai trò là giao diện giữa người dùng và máy tính.
- Hai mục tiêu chính :
  - Để sử dụng hệ thống máy tính một cách dễ dàng
  - Để sử dụng hệ thống tài nguyên máy tính một cách có hiệu quả

Giao tiếp giữa người dùng và máy tính

- Nếu xây dựng một ứng dụng dưới dạng một tập các chỉ thị máy (ngôn ngữ máy)→ phức tạp
- Để giảm bớt sự phức tạp→ cung cấp một tập các system program. Một số system program được xem như các tiện ích. Xây dựng các hàm công cụ được dùng thường xuyên, trợ giúp trong tạo chương trình, quản lý tập tin, thư mục và điều khiển các thiết bị I/O.
- System program quan trọng nhất đó là hệ điều hành

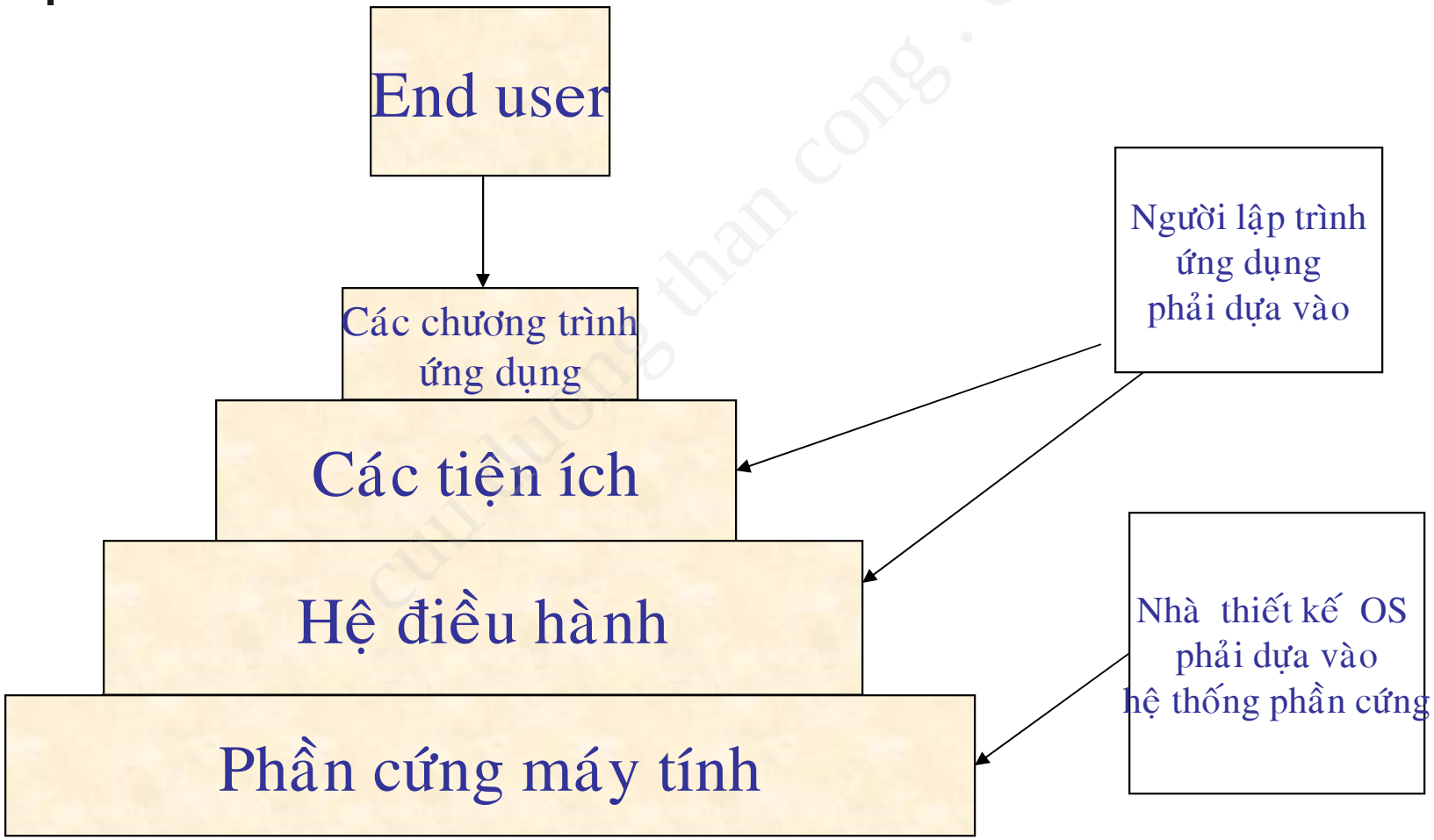
Giao tiếp giữa người dùng và...

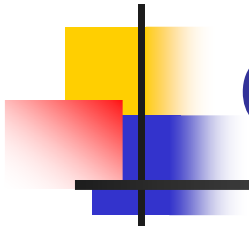
- OS che các chi tiết phần cứng bên dưới và cung cấp một giao diện thuận tiện để sử dụng
- OS cung cấp dịch vụ theo các hướng
  - Tạo chương trình: phương tiện, dịch vụ hỗ trợ
  - Thực thi chương trình: chuẩn bị, nạp, khởi động, cấp phát

Giao tiếp giữa người dùng và...

- .....
  - Truy xuất thiết bị I/O : I/O ⊃ Chỉ thị, tín hiệu điều khiển. OS (Chỉ thị , điều khiển)→đọc, ghi
  - Truy xuất tập tin: Thống nhất mọi cách đọc ghi file với tất cả các loại thiết bị. Cấp cơ chế bảo vệ, chia sẽ.
  - Truy xuất hệ thống: điều khiển đăng nhập.

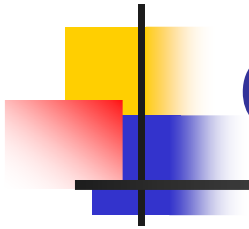
Giao tiếp giữa người dùng và..





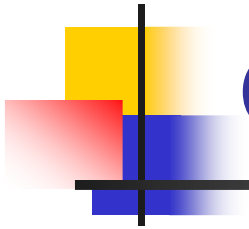
Quản lý tài nguyên

- Tài nguyên dùng để di chuyển, lưu trữ, xử lý dữ liệu và điều khiển các chức năng này. OS chịu trách nhiệm quản lý các tài nguyên này.
- OS điều khiển các chức năng cơ bản của máy tính, nhưng có đặc thù:
  - Các chức năng của OS tương tự như phần mềm máy tính thông thường, đó là được thực thi bởi CPU
  - OS thường xuyên nhường điều khiển và phải lệ thuộc vào CPU trong việc lấy lại điều khiển.



Quản lý tài nguyên

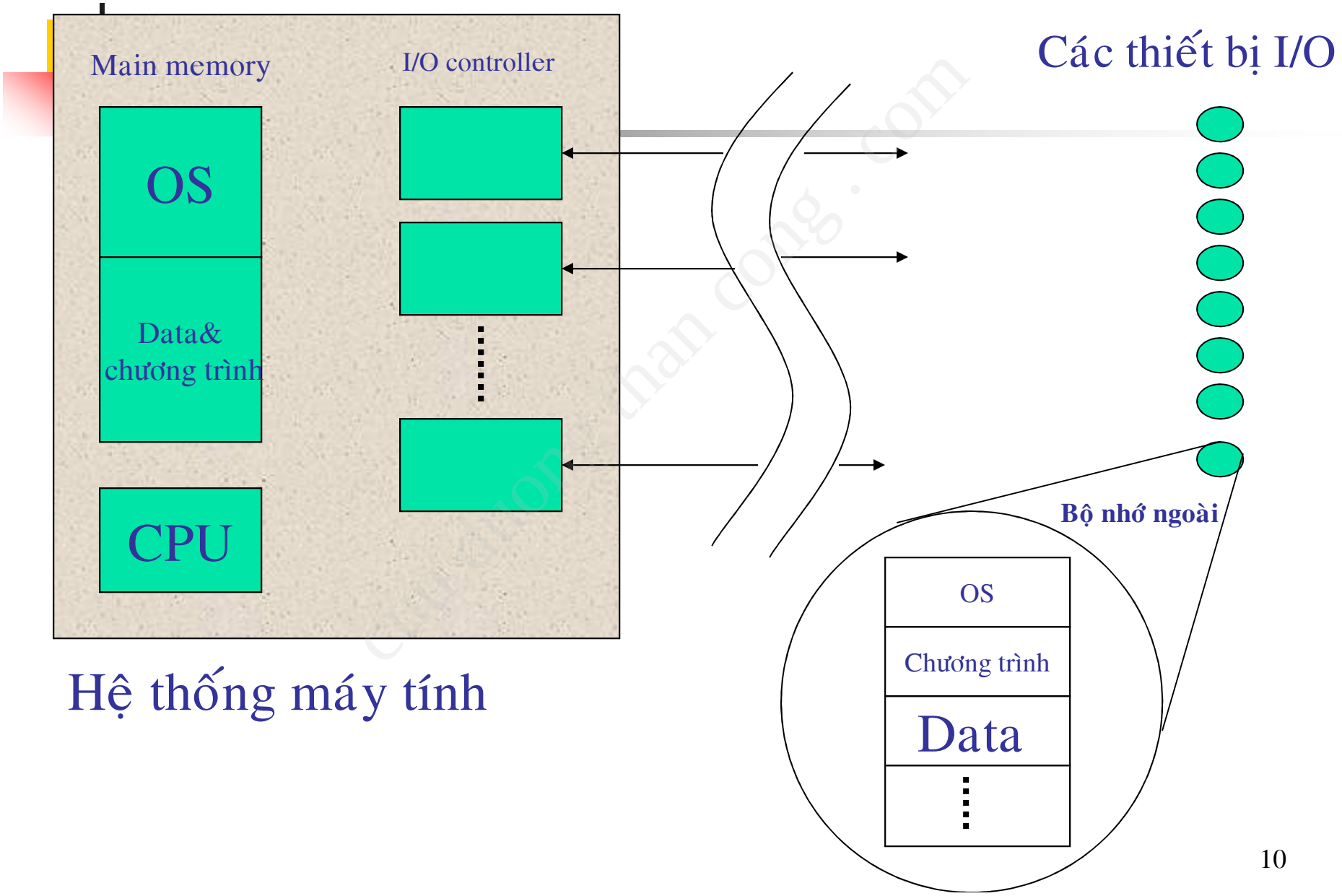
- Thực tế OS không khác gì hơn là một chương trình máy tính
- Chỉ khác mục đích
  - Hướng dẫn CPU sử dụng các tài nguyên khác nhau
  - Định thời gian thực thi các chương trình khác nhau



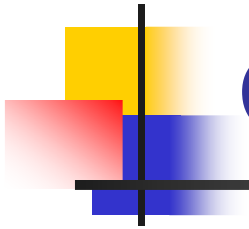
Quản lý tài nguyên

- Một phần của OS ở trong bộ nhớ chính gọi là "nhân" (kernel, core, nucleus), chứa hầu hết các chức năng được dùng thường xuyên nhất. Phần còn lại thường được gọi là vỏ (shell) được lưu giữ trên bộ nhớ ngoài, khi cần được lấy vào.
- Phần còn lại của bộ nhớ chính chứa dữ liệu và chương trình khác. Việc phân phối bộ nhớ chính được thực hiện trong sự phối hợp điều khiển giữa OS và phần cứng quản lý bộ nhớ (memory circuits)
- OS quyết định khi nào chương trình có thể sử dụng thiết bị I/O để thực thi và điều khiển truy xuất, sử dụng tập tin.

Quản lý tài nguyên...



Hệ thống máy tính



Các loại hệ điều hành

- Phân loại dựa vào các đặc tính chủ yếu. Có hai khuynh hướng phân loại độc lập nhau:
  - Hệ thống xử lý lô (batch) hay giao tác
  - Đơn chương hay đa chương

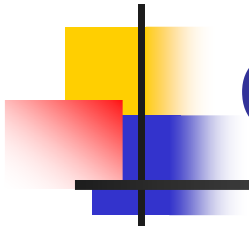
Các loại hệ điều hành...

- Hệ thống giao tác (interactive system): người dùng giao tiếp trực tiếp với máy tính để yêu cầu thực hiện một công việc hay một giao tác nào đó
- Hệ thống lô (batch system): chương trình của user được gộp lại với những chương trình của các user khác, giao cho điều hành viên máy tính để điều hành viên này cho máy thực hiện. Sau đó, kết quả sẽ được giao lại cho từng user. Các hệ thống lô ngày nay rất hiếm. Tuy nhiên hệ thống này vẫn còn giá trị mô tả.

Các loại hệ điều hành...

- Hệ thống đơn chương (uniprogramming): Chỉ có một chương trình được nạp vào bộ nhớ và xử lý cho đến khi kết thúc. Vào một thời điểm chỉ có một chương trình đang chạy.
- Hệ thống đa chương (multiprogramming): cố gắng tận dụng tối đa năng lực của CPU, có nhiều chương trình làm việc vào một thời điểm.

  CPU chuyển phục vụ trong số các chương trình được nạp vào trong bộ nhớ

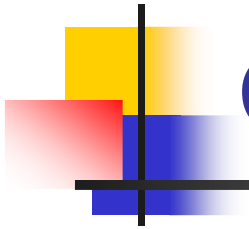


Các hệ thống thời kỳ đầu

- Các máy tính thời kỳ đầu, người lập trình phải thao tác trực tiếp với các thành phần phần cứng. Các mã chương trình được nạp vào qua thiết bị nhập, ví dụ card reader.
- Các hệ thống thời kỳ đầu tồn tại hai vấn đề chính
  - Lập lịch: sử dụng bảng đăng ký giờ cố định
  - Thời gian cài đặt: một chương trình đơn (một job) liên quan đến nạp trình biên dịch, lưu giữ chương trình đối tượng, nạp và liên kết với các hàm. Mỗi bước đều tham chiếu đến băng từ.

Các hệ thống lô đơn giản

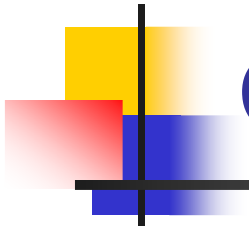
- Để khắc phục hai vấn đề trên, hệ điều hành xử lý lô đơn giản đã được phát triển. Trở thành monitor. User không mất nhiều thời gian để truy xuất trực tiếp máy. Thay vì vậy user ghi job lên card hay băng từ rồi giao cho điều hành viên
- Monitor thường trú trong bộ nhớ. Monitor đọc job đặt vào vùng chương trình user trong bộ nhớ. Điều khiển được chuyển cho job này. Khi kết thúc job một ngắt xảy ra chuyển điều khiển cho monitor. Monitor nhanh chóng đọc job kế tiếp.



Các hệ thống lô ....

- Vào thời điểm nào đó, CPU thực thi phần bộ nhớ chứa monitor. Các chỉ thị khiến cho job kế được đọc vào vùng nhớ khác. Khi job đã được đọc, CPU sẽ gặp phải một chỉ thị bảo CPU thực thi vùng nhớ có chứa job. CPU tiếp tục thực hiện chương trình cho đến khi kết thúc hay gặp điều kiện lỗi. Cả hai sự kiện đều khiến cho CPU nạp chỉ thị kế tiếp trong monitor
- "chuyển điều khiển cho job"...
- "trả điều khiển về cho monitor"...

Monitor thường trú trong bộ nhớ



Các hệ thống lô ....

- Monitor kiểm soát vấn đề lập lịch:
- Monitor kiểm soát vấn đề cài đặt: các chỉ thị được bao hàm ngôn ngữ điều khiển. Ngôn ngữ điều khiển chỉ là một loại ngôn ngữ lập trình nhằm cung cấp các chỉ dẫn cho monitor
- Monitor chỉ là một chương trình máy tính, dựa vào khả năng của CPU để luân phiên bắt lấy và phóng thích điều khiển

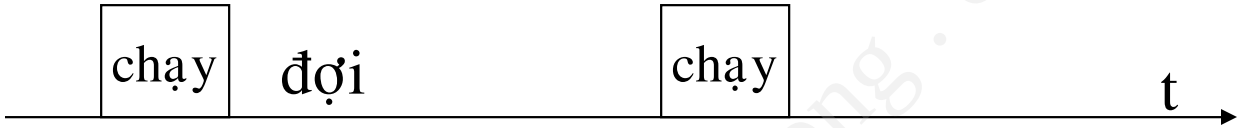
Các hệ thống lô ....

- Một số các đặc tính phần cứng cũng được yêu cầu:
  - Bảo vệ bộ nhớ
  - Định thời
  - Các chỉ thị có đặc quyền, chỉ được thực thi bởi monitor. Ví dụ các chỉ thị I/O

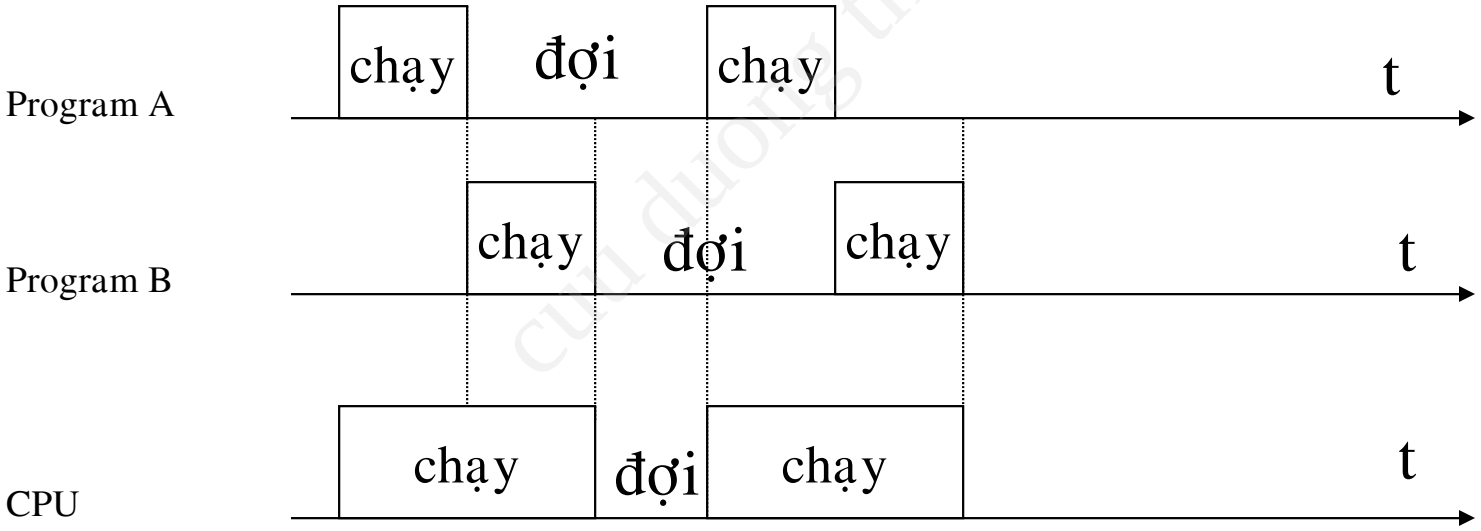
Các hệ thống lô phức tạp

- Truy xuất I/O chậm cho dù hệ thống lô đơn giản đã tự động hóa nạp job, lãng phí CPU=> Cần hệ thống lô phức tạp tận dụng tối đa công năng của CPU.
- Bộ nhớ có thể chứa cùng lúc nhiều job.
- Khi một job phải đợi I/O thì CPU chuyển sang job khác không đợi I/O. Quá trình này được gọi là đa chương (multiprogramming)

Các hệ thống lô phức tạp...



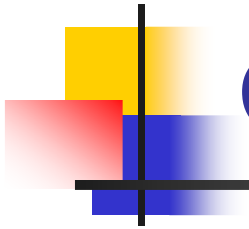
(uniprogramming)



(multiprogramming)

Các hệ thống lô phức tạp...

- Hệ thống đa chương phải dựa trên các đặc tính phần cứng nào đó.
- Hầu hết các đặc tính phần cứng bổ sung là I/O interrupt và DMA.
- Cần phải kiểm soát nội dung trong bộ nhớ=>quản lý bộ nhớ (memory management)
- Nếu nhiều job đều ở trạng thái sẵn sàng cần quyết định chọn job nào=>lập lịch (scheduling)

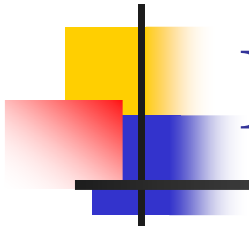


Chia sẻ thời gian

- Xử lý lô đa chương giúp sử dụng CPU hiệu quả. Tuy nhiên, có một số ứng dụng khác thì điều mong muốn chủ yếu lại là cung cấp cho user khả năng tương tác trực tiếp với máy tính.
- Ngày nay yêu cầu tính toán tương tác có thể được đáp ứng bởi các microcomputer chức năng và kỹ thuật chia sẻ thời gian
- Kỹ thuật chia sẻ thời gian có thể được dùng để kiểm soát nhiều job giao tác => Thời gian của CPU được chia sẻ cho nhiều user.
- Xử lý lô phức tạp và chia sẻ thời gian đều sử dụng đa chương.

So sánh giữa HĐH xử lý lô đa chương và HĐH chia sẻ thời gian

	Xử lý lô đa chương	Chia sẻ thời gian
Mục đích	Tận dụng tối đa công năng của CPU	Đáp ứng nhanh
Nguồn chỉ thị	Các chỉ thị ngôn ngữ điều khiển được cung cấp cùng với job	Các lệnh được nhập từ đầu cuối

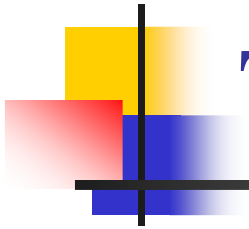


Bài 2_Tổ chức và Hoạt động

- Tiến trình
- Lập lịch
- Quản lý bộ nhớ
- Bộ nhớ ảo

Tiến trình (process)

- Lý thuyết HĐH được phát triển với thuật ngữ process thay cho job. Nhiều định nghĩa về process đã được công bố:
  - Là một chương trình đang thực thi
  - "Linh hồn sống động" của một chương trình
  - Một thực thể chiếm giữ bộ xử lý
- Ở đây ta sử dụng định nghĩa thứ nhất: Process là một chương trình user đang trong tiến trình thực thi, liên kết với một không gian địa chỉ. Cũng liên kết với các thanh ghi PC,SP(stack pointer), một số thanh ghi và tất cả các thông tin cần thiết khác.



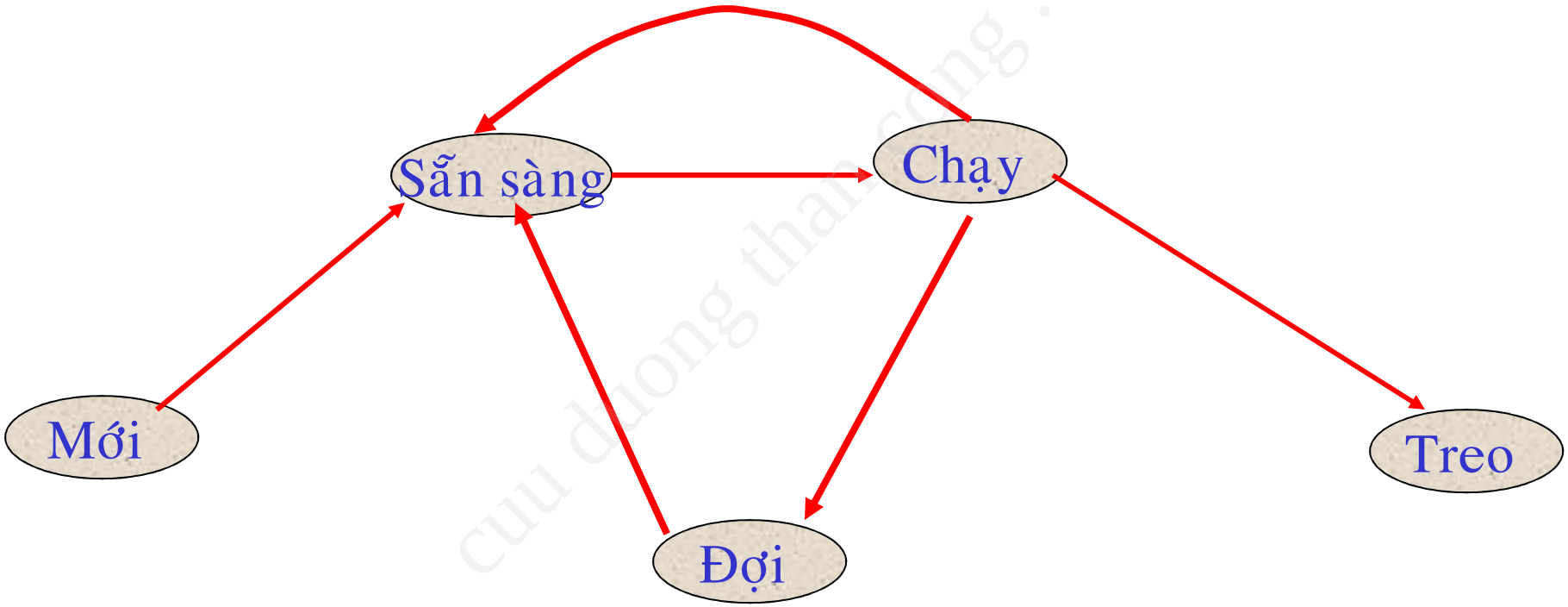
Tiến trình...

- KGĐC chứa Data, chương trình,stack=> được gọi là core image.
- Tất cả các thông tin về mỗi process khác với core image được lưu trong process table
- Vậy:

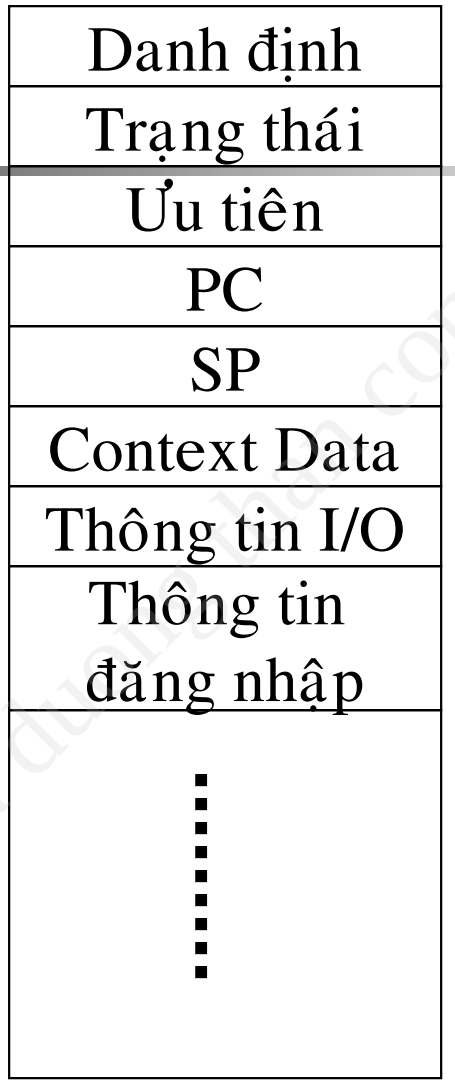
  Process = core image + process table

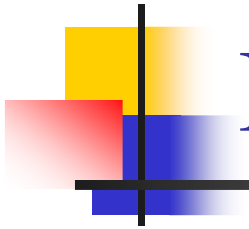
Các trạng thái của process

- Trong thời gian "sống" của process, trạng thái của nó thay đổi một số lần. Trạng thái của một process được xác định và gắn liền với thông tin tồn tại tương ứng. Có năm trạng thái có thể của một process:
  - Mới: một chương trình được kết nạp bởi bộ lập lịch nhưng chưa sẵn sàng hoạt động.
  - Sẵn sàng: sẵn sàng hoạt động và đang đợi CPU
  - Chạy : đang được CPU xử lý
  - Đợi :đang treo tạm thời giữa lúc được thực thi để chờ tài nguyên
  - Treo : quá trình đã kết thúc và sẽ bị hủy bởi HĐH



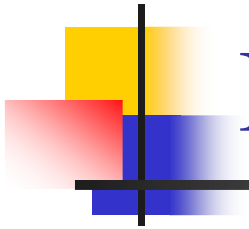
PROCESS TABLE





Lập lịch

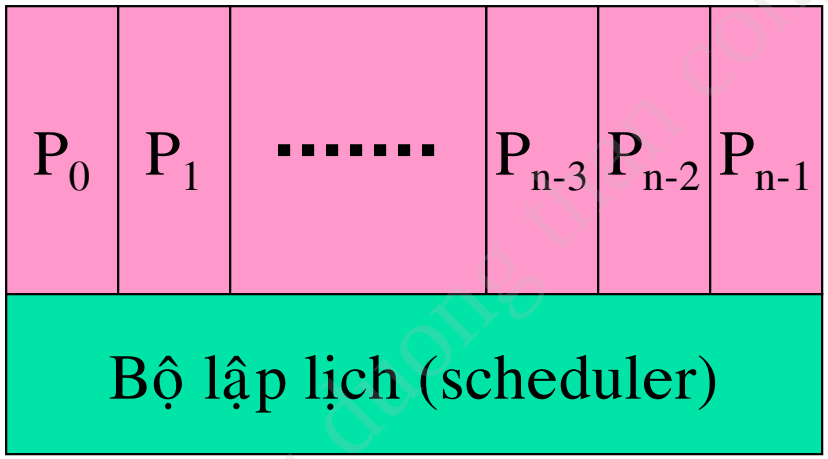
- Các hệ điều hành hiện đại đều là đa chương, đây cũng là một nhiệm vụ trọng tâm.
- Đa chương thì có nhiều job được duy trì trong bộ nhớ, các job luân phiên nhau chiếm dụng CPU rồi đợi I/O.
- Bộ xử lý luôn bận rộn bởi thực thi một job trong khi các job khác đang đợi.
- Bí quyết của đa chương là lập lịch.

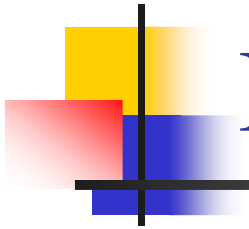


Lập lịch...

- Lớp thấp nhất của OS là bộ lập lịch (scheduler), tất cả kiểm soát ngắt và khởi động/dừng các quá trình thuộc về scheduler.
- Phần còn lại của OS được cấu thành từ các process

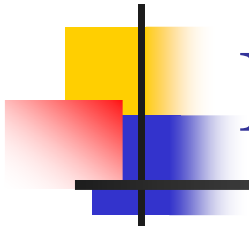
Lập lịch...





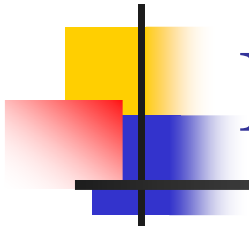
Lập lịch..

- Lập lịch mức cao
  - Bộ lập lịch xác định chương trình nào được nạp vào hệ thống để trở thành một process.
  - Kiểm soát mức độ đa chương (số process có thể hiện hữu trong bộ nhớ).
  - Càng nhiều process trong bộ nhớ→ lượng thời gian được CPU phục vụ mỗi lần cho các process càng nhỏ.
  - Trong hệ thống lô các job mới đến được giữ trong hàng đợi. Bộ lập lịch mức cao sẽ thêm job vào hệ thống từ hàng đợi khi thấy có thể. Có hai quyết định
    - Lấy một hay nhiều job
    - Job nào sẽ được lấy



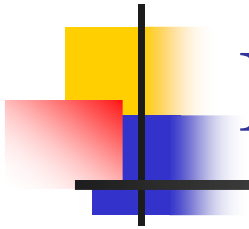
Lập lịch..

- Lập lịch mức cao...
  - Trong hệ thống chia sẻ thời gian request process được phát ra từ nỗ lực kết nối của user.
    - Không xếp hàng và giữ các quá trình trong hàng đợi một cách đơn giản như hệ thống lô.
    - OS chấp nhận tất cả những yêu cầu hợp lệ cho đến khi hệ thống hết khả năng.



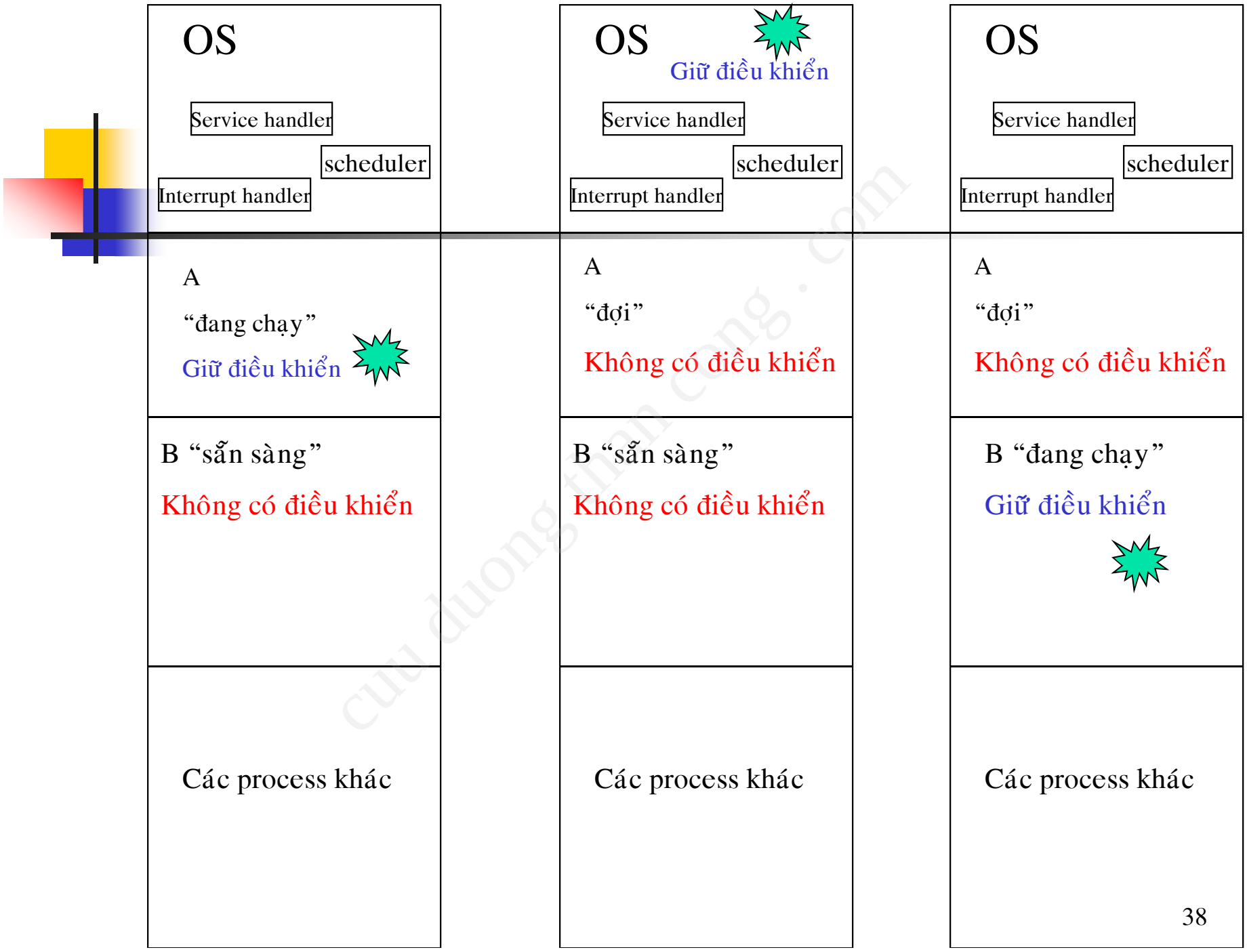
Lập lịch..

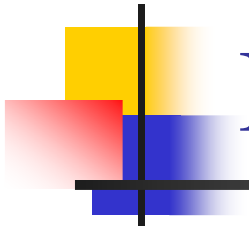
- Lập lịch vận hành
  - Bộ lập lịch vận hành thực hiện thường xuyên
  - Đưa ra quyết định job nào sẽ là kế tiếp
  - Được gọi là short-term scheduler hay dispatcher
- Thực hiện lập lịch
  - Ví dụ hiện có một số các process đang hoạt động cùng với process A và process B. vào thời điểm bắt đầu CPU đang phục vụ A, sau đó CPU bị chiếm chuyển sang thực thi OS vì một trong ba lý do:
    - A phát ra một lời gọi dịch vụ (ví dụ yêu cầu I/O) đến OS
    - A gây ra một ngắt cứng đến CPU và CPU chuyển sang phục vụ cho Interrupt handler trong OS
    - Các sự kiện khác cần phục vụ phát ra một ngắt



Lập lịch...

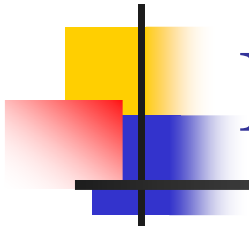
- Thực hiện lập lịch...
  - Trong bất kỳ trường hợp nào ngữ cảnh hiện hành của A đều được lưu giữ và CPU bắt đầu thực thi trong OS→ OS thực thi một số việc (ví dụ khởi động I/O)→bộ lập lịch vận hành quyết định process nào sẽ được thực hiện trước. Trong ví dụ này là B.→ OS hướng dẫn CPU nạp ngữ cảnh của B và thực thi





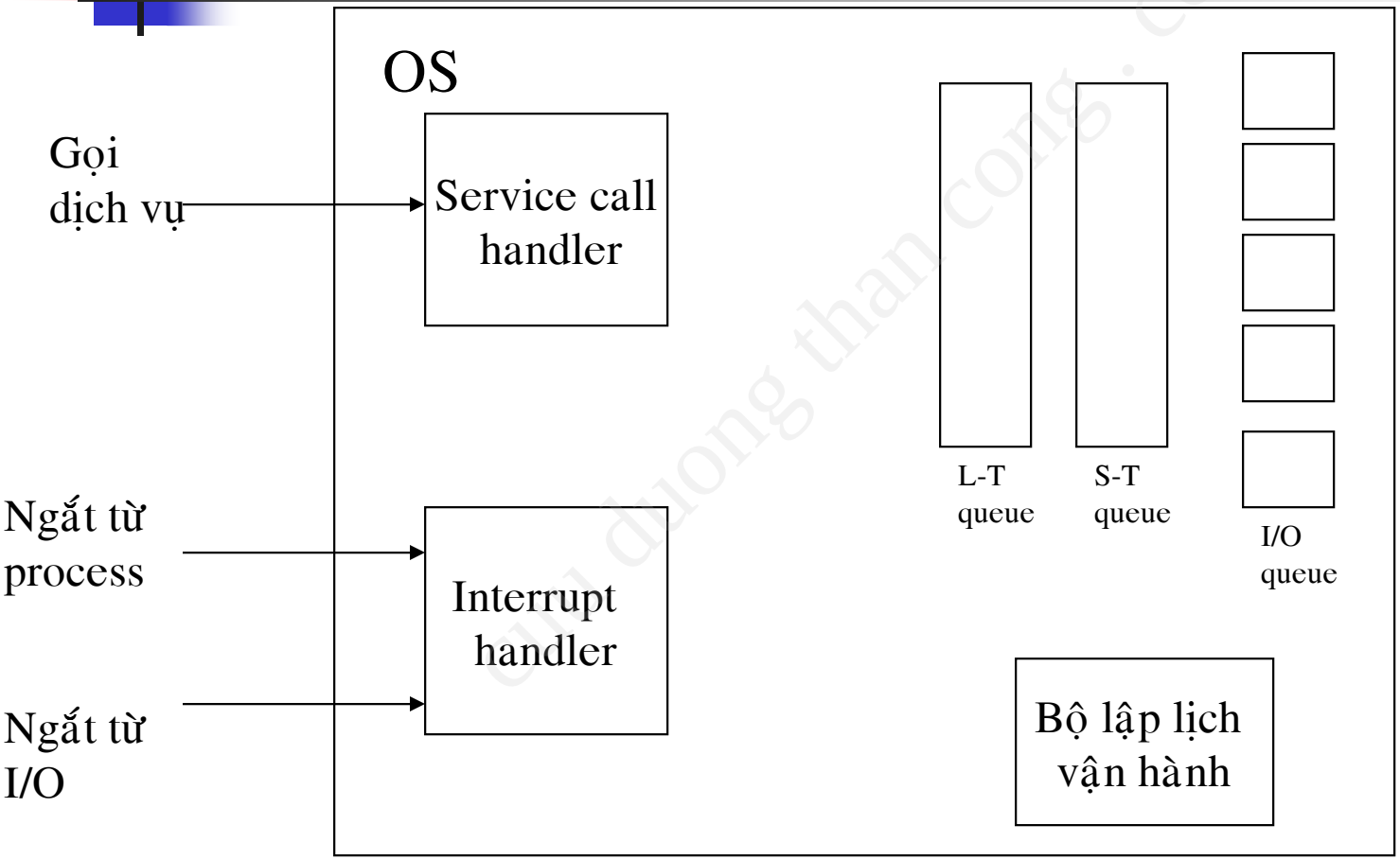
Lập lịch...

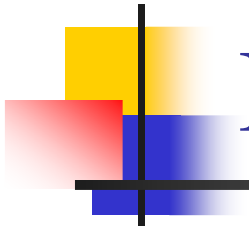
- OS tiếp nhận control tại các handler, khi ngắt hay dịch vụ được gọi thì bộ lập lịch vận hành hoạt động.
- OS duy trì một số hàng đợi. Mỗi hàng đợi là một danh sách các quá trình đang đợi tài nguyên.
  - Long-term queue: danh sách các job đang chờ dùng hệ thống. Khi điều kiện cho phép bộ lập lịch mức cao cấp phát bộ nhớ và tạo ra quá trình cho job
  - Short-term queue: danh sách các quá trình ở trạng thái sẵn sàng. Quyết định chọn là do bộ lập lịch vận hành. Bộ lập lịch vận hành sẽ làm việc dựa vào một thuật toán lập lịch nào đó, ví dụ Round-Robin, ưu tiên, nhiều hàng đợi, shortest process next, fair-share...



Lập lịch...

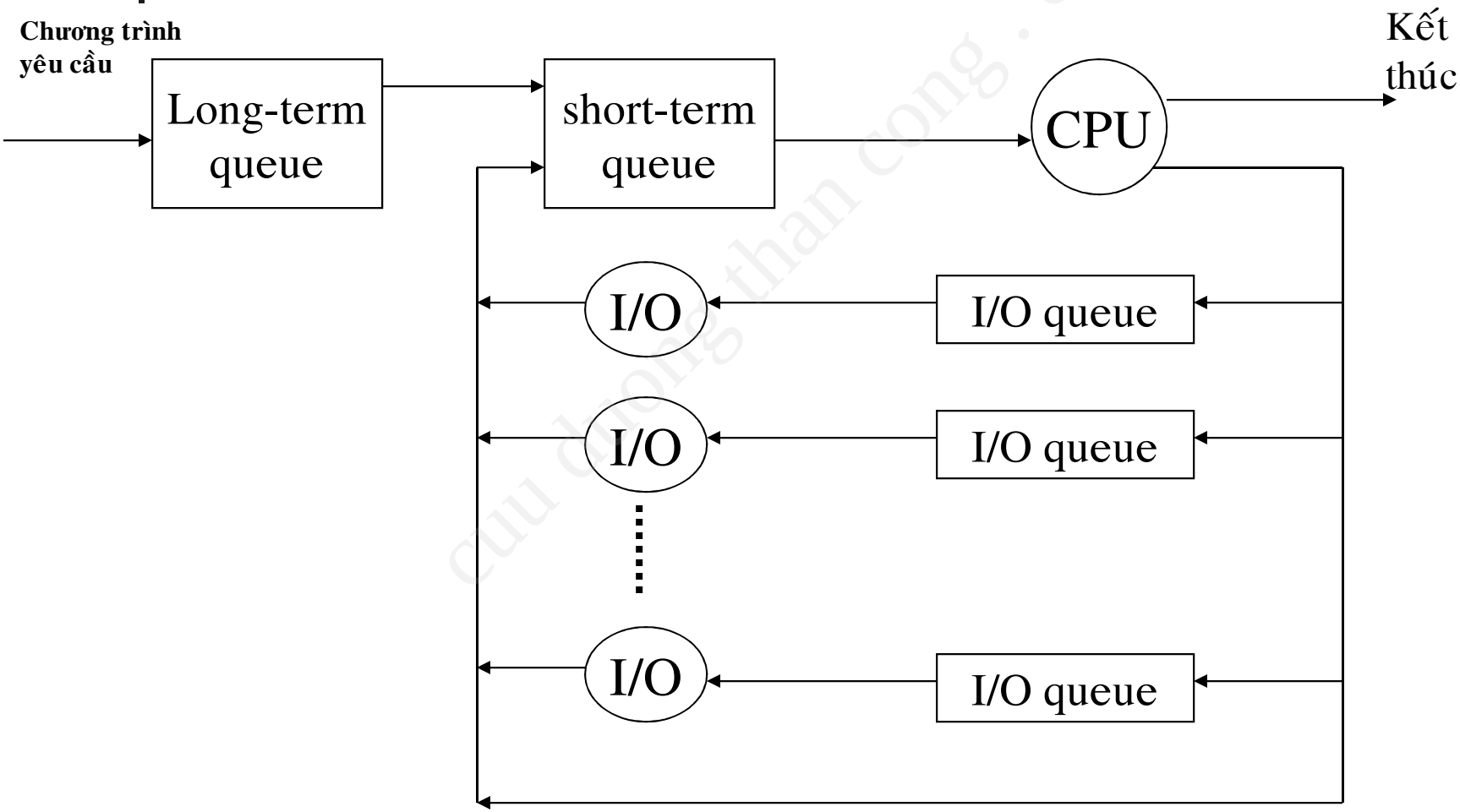
- OS duy trì một...
  - I/O queue cho mỗi thiết bị I/O. Có thể có nhiều process cùng yêu cầu một I/O, tất cả các quá trình yêu cầu I/O được xếp hàng

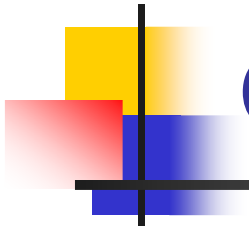




Lập lịch....

- Nếu quá trình đang "chạy" bị treo tạm thời bởi yêu cầu I/O, nó được đặt vào hàng đợi I/O tương ứng.
- Nếu bị treo do timeout hay bởi OS lấy điều khiển thì process được đặt vào trạng thái "sẵn sàng" và đưa ngược trở về hàng đợi short-term
- Khi hoạt động I/O hoàn tất, OS xóa bỏ process khỏi hàng đợi I/O và đặt nó vào hàng đợi short-term. Sau đó OS chọn process khác đang đợi I/O và phát tín hiệu đến thiết bị I/O để đáp ứng yêu cầu của process này.





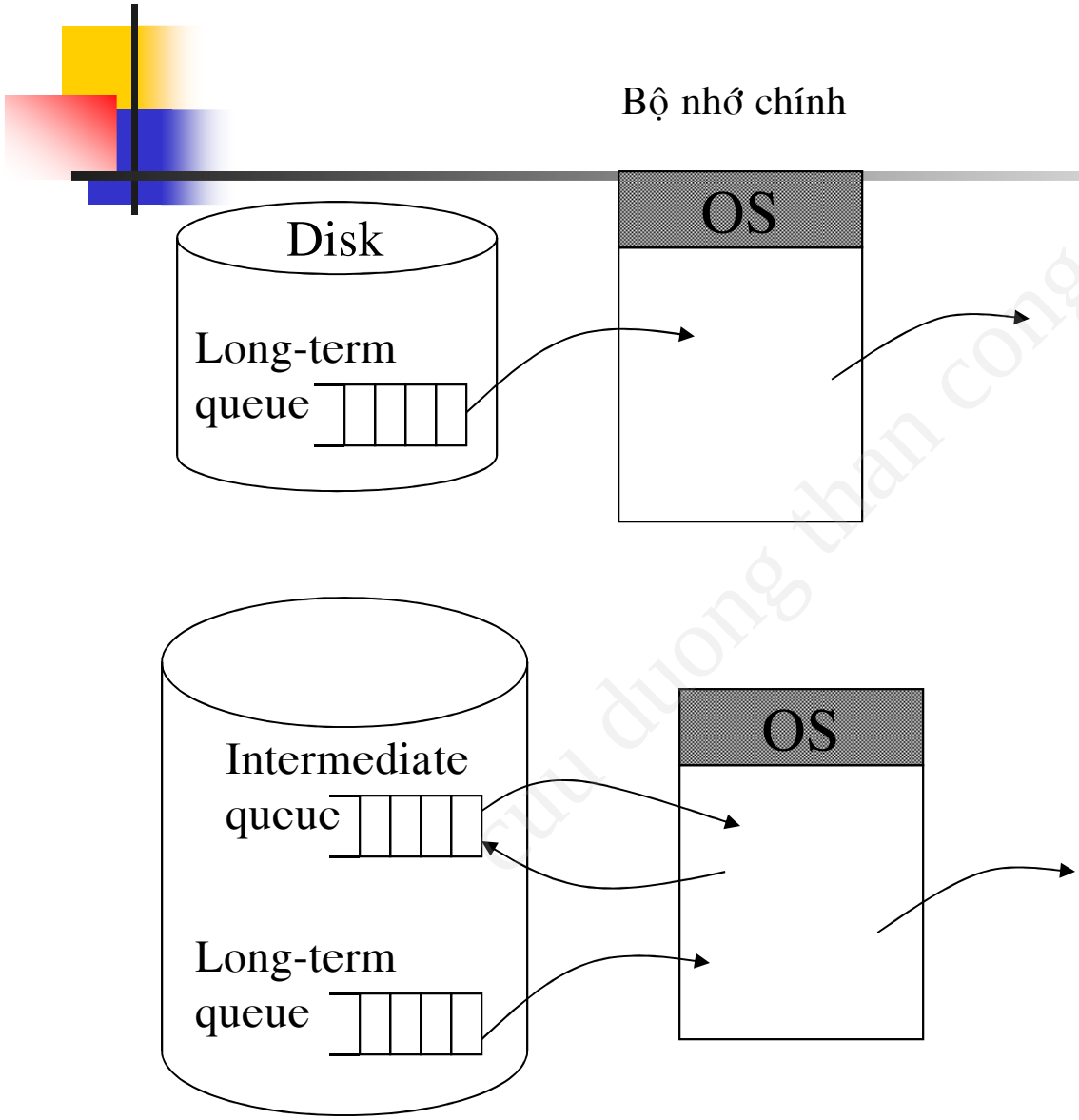
Quản lý bộ nhớ

- Trong hệ thống đa chương một phần bộ nhớ dành cho chương trình user cần được chia ra để chứa nhiều chương trình một lúc. Công tác phân chia này được thực hiện bởi hệ điều hành và được gọi là **quản lý bộ nhớ.**
- Cần phân phối bộ nhớ một cách có hiệu quả sao cho chứa được nhiều process

Quản lý bộ nhớ-Swapping

- Tổ chức lập lịch vào các hàng đợi vẫn không tận dụng hết được khả năng của CPU ?
- Giải pháp mở rộng bộ nhớ để chứa được nhiều process hơn. Nhưng có hai vấn đề tồn tại:
  - Bộ nhớ đắt tiền
  - Chương trình trong bộ nhớ sẽ lớn nếu giá bộ nhớ giảm

  =>bộ nhớ càng lớn thì process càng lớn
- Giải pháp khác là swapping (hoán đổi). Long-term queue được lưu trên đĩa cứng. Process được mang vào khi có không gian khả dụng. Khi process được hoàn tất sẽ được loại khỏi bộ nhớ. Lúc này không có process nào trong bộ nhớ là ở trạng thái sẵn sàng. Chúng ở trạng thái nhàn rỗi, bộ xử lý sẽ swap một trong các process này ra hàng đợi trung gian (intermediate queue). OS lại mang process khác từ hàng đợi trung gian hay từ long-term vào bộ nhớ.

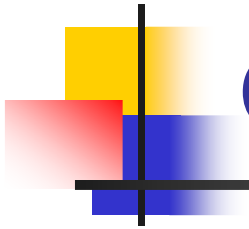


Quản lý bộ nhớ-Swapping

- Swapping là một hoạt động I/O
- Hard disk thường được dùng vì là I/O nhanh nhất
- Kết hợp với bộ nhớ ảo để cải thiện hiệu suất.

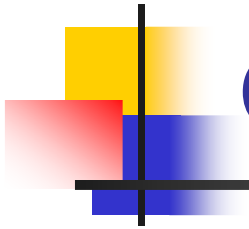
Quản lý bộ nhớ-Partitioning

- Phần còn lại của bộ nhớ được phân hoạch để nhiều process sử dụng. Các phân hoạh đơn giản nhất là chia cố định, các phần có thể có kích thước không bằng nhau.
- Lãng phí bộ nhớ
- Giải pháp hiệu quả hơn là phân hoạch động, cấp lượng bộ nhớ đúng bằng kích thước process.
- Trong quá trình hoạt động, sẽ xuất hiện các "lỗ trống" => dùng kỹ thuật kết chặt (compaction)



Quản lý bộ nhớ-Partitioning

- Process cư ngụ trong bộ nhớ có chỉ thị và dữ liệu được địa chỉ hóa. Nếu phân hoạch cố định điều này không là vấn đề, nếu phân hoạch động→ vấn đề phát sinh?
- Khái niệm địa chỉ luận lý xuất hiện: địa chỉ luận lý được biểu diễn như một vị trí liên hệ với điểm bắt đầu của chương trình. Các chỉ thị trong chương trình chỉ chứa địa chỉ luận lý.
- Khi bộ xử lý thực thi một process nó tự động chuyển đổi từ địa chỉ luận lý về địa chỉ vật lý bằng cách cộng vị trí hiện hành của process (gọi là địa chỉ nền) với mỗi địa chỉ luận lý.



Quản lý bộ nhớ-Partitioning

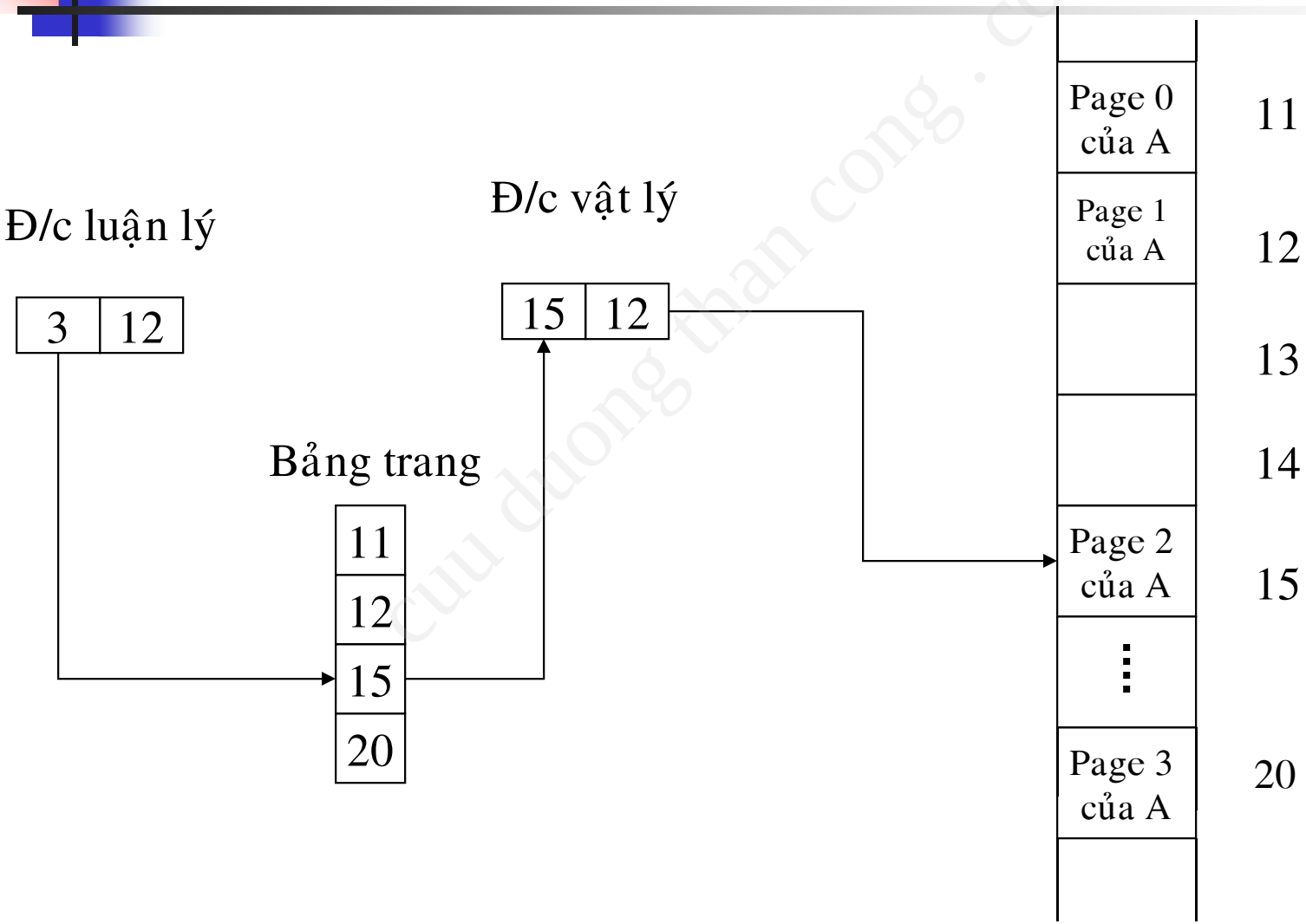
- Chuyển đổi địa chỉ là một đặc trưng của CPU nhằm phù hợp với các yêu cầu của OS
- Bản chất của hệ thống phần cứng phụ thuộc vào chiến lược quản lý bộ nhớ sẽ được dùng trong OS

Quản lý bộ nhớ-Paging

- Cả phân hoạch cố định và động ở trên đều không hiệu quả trong việc sử dụng bộ nhớ.
- Có thể phân hoạch cố định bộ nhớ thành các phần có kích thước nhỏ và bằng nhau, mỗi phần như vậy được gọi là ***frame***. Mỗi process được chia thành một số phần có kích thước bằng nhau, các phần như vậy được gọi là các ***trang (page).*** Mỗi trang được nạp vào đúng một frame, được gọi là ***page frame*** (frame chứa trang).
- Phần bộ nhớ còn dư trong mỗi phần chia cho process sẽ thuộc về trang sau cùng.

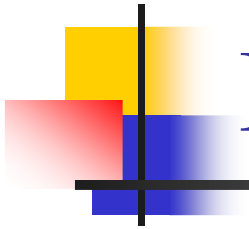
Quản lý bộ nhớ-Paging

- Nếu cần nạp một process khác mà không có đủ số frame liên tục thì sao?
- Hệ điều hành duy trì một ***"bảng trang"*** cho mỗi process. Bảng trang chỉ ra vị trí frame của mỗi trang chứa process
- Trong chương trình mỗi địa chỉ luận lý gồm có chỉ số trang và địa chỉ tương quan với trang đó.
- CPU phải biết cách truy xuất bảng trang của process hiện hành và dùng nó để tạo ra các địa chỉ vật lý.



Quản lý bộ nhớ-Paging

- Giải pháp phân trang giải quyết được các vấn đề tồn tại từ trước?
- Bộ nhớ chính được chia thành nhiều frame nhỏ có kích thước bằng nhau. Mỗi process có kích thước nhỏ hơn sẽ cần số frame ít hơn và ngược lại.
- Khi process được nạp vào, các trang của nó được nạp vào các frame khả dụng và một bảng trang được xây dựng.

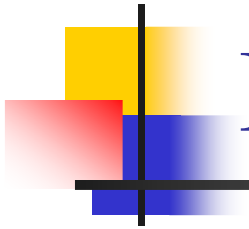


Bộ nhớ ảo

- Sử dụng phân trang giúp cho đa chương thực sự hiệu quả.
- Còn phát triển khái niệm quan trọng khác: bộ nhớ ảo
- Phân trang theo nhu cầu (demand paging) là một cải tiến tinh tế trong lược đồ phân trang. Mỗi trang của process được mang vào chỉ khi thật sự cần thiết. Tại sao như vậy?

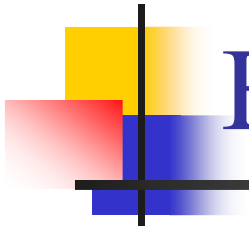
Bộ nhớ ảo...

- Khi chương trình rẽ nhánh đến một chỉ thị (hay data) khác trên một trang mà không có trong bộ nhớ → một sự kiện lỗi trang (page fault) được tạo ra bảo OS mang trang mong muốn vào bộ nhớ.
- *"Có nhiều process"* trong bộ nhớ hơn?
- Tiết kiệm thời gian swapping?
- Tuy nhiên, OS phải khéo léo quản lý lược đồ này. Khi mang một trang vào, một trang khác phải bị bỏ ra.
- Nếu một trang bị loại ra ngay trước khi được sử dụng thì ngay sau đó lại phải nạp vào. Nếu quá nhiều tình huống như vậy sẽ dẫn hệ thống đến tình trạng gọi là ***thrashing.***



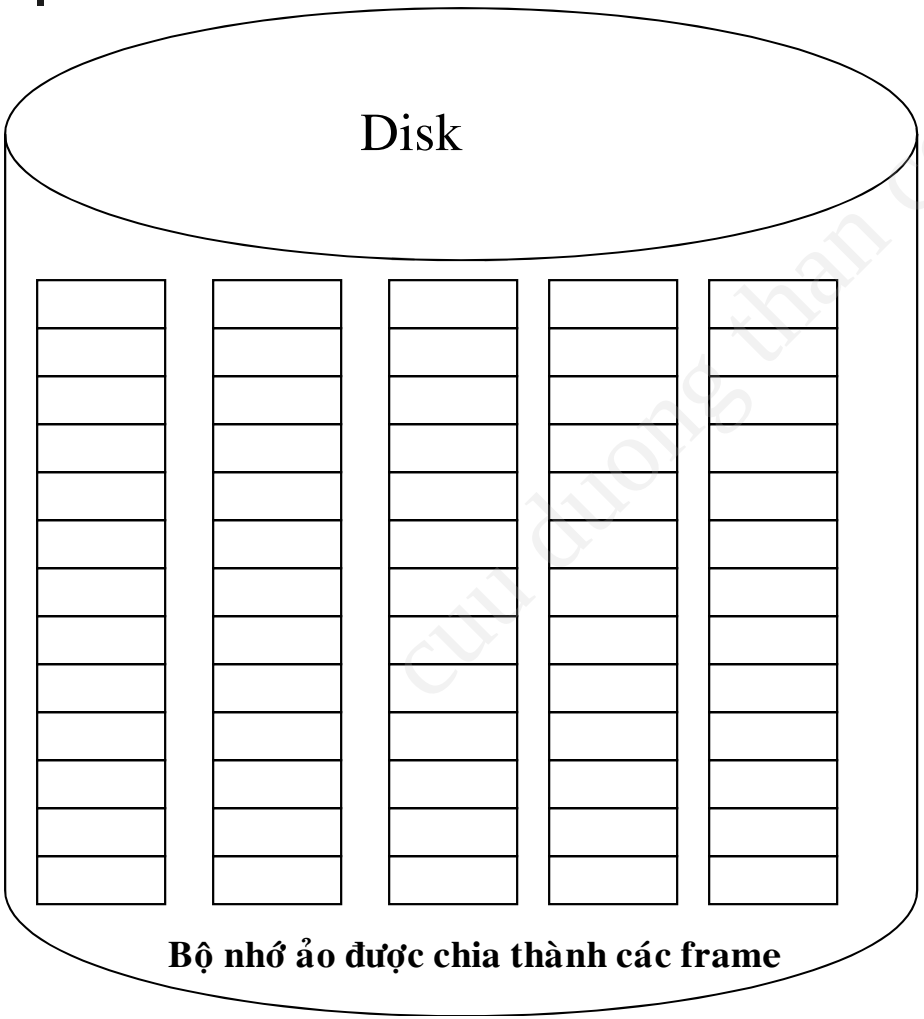
Bộ nhớ ảo...

- Hệ điều hành dựa vào những dữ kiện quá khứ gần để cố gắng đoán xem trang nào sẽ có ít khả năng được dùng trong tương lai gần.
- Với phân trang theo yêu cầu hệ thống không cần nạp toàn bộ process vào trong bộ nhớ. Điều này dẫn đến một kết quả rất đặc biệt trong sử dụng máy tính: xóa bỏ một trong những hạn chế cơ bản nhất trong lập trình.

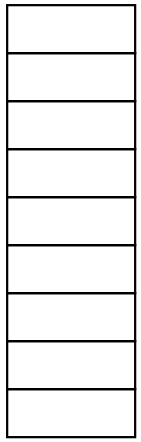


Bộ nhớ ảo...

- Ở chừng mực nào đó, người lập trình sẽ làm việc với một bộ nhớ chính rộng hơn thực sự.
- Vì các process chỉ thực thi trong bộ nhớ chính nên bộ nhờ này còn gọi là bộ nhớ thực. Nhưng người lập trình lại làm việc với một bộ nhớ rộng hơn nhiều (được phân phối ngay trên đĩa cứng), bộ nhớ rộng này được gọi là bộ nhớ ảo.
- Bộ nhớ ảo cho phép đa chương một cách hiệu quả và giải phóng người dùng khỏi các ràng buộc không cần thiết của bộ nhớ chính.



Bộ nhớ chính



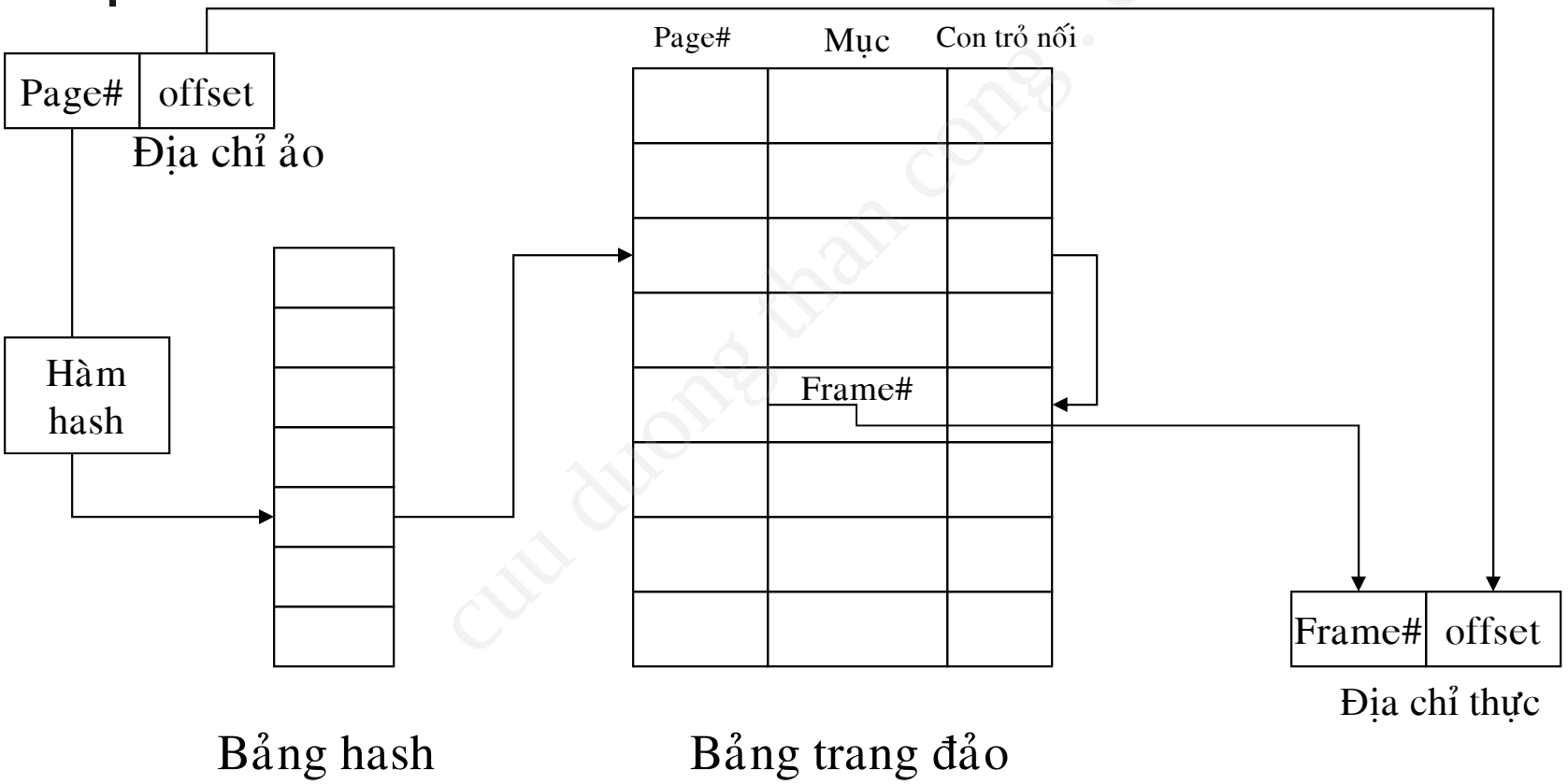
Các frame thực

Bảng trang

- Cơ cấu cơ bản để đọc một từ nhớ liên quan đến công việc chuyển đổi địa chỉ luận lý (chỉ số trang và offset) thành địa chỉ vật lý (chỉ số frame và offset) sử dụng bảng trang.
- Bảng trang có thể rất lớn nên phải lưu trên bộ nhớ ảo. Như vậy bảng trang phải chịu phân trang như các trang khác.
- Khi process đang chạy, ít nhất phải có một phần của bảng trang trong bộ nhớ chính, chứa các mục liên quan đến các trang thực thi hiện hành.
- Một số bộ xử lý dùng lược đồ đa mức để tổ chức bảng trang lớn hơn. Trong đó có thư mục chỉ đến các bảng trang khác (Pentium).
- Nếu chiều dài của thư mục là X và chiều dài tối đa của bảng trang là Y thì process có đến X.Y trang. Thông thường kích thước của bảng trang được giới hạn bằng kích thước của một trang.

Bảng trang

- Một giải pháp khác để tổ chức trang đa mức là dùng cấu trúc bảng trang đảo (máy tính của IBM).
  - Phần chỉ số trang trong địa chỉ ảo ánh xạ sang một *bảng hash* dùng một *hàm hash* đơn giản. Bảng hash chứa con trỏ chỉ đến *bảng trang đảo*, ở đó chứa các mục thông tin về trang. Như vậy chỉ có một mục trong bảng hash và bảng trang đảo cho một frame bộ nhớ thực, thay vì một frame ảo.
  - Có một phần cố định trong bộ nhớ thực cho mỗi bảng bất chấp số process hay số trang ảo được hỗ trợ.



Bộ đệm TLB (Translation Lookaside Buffer)

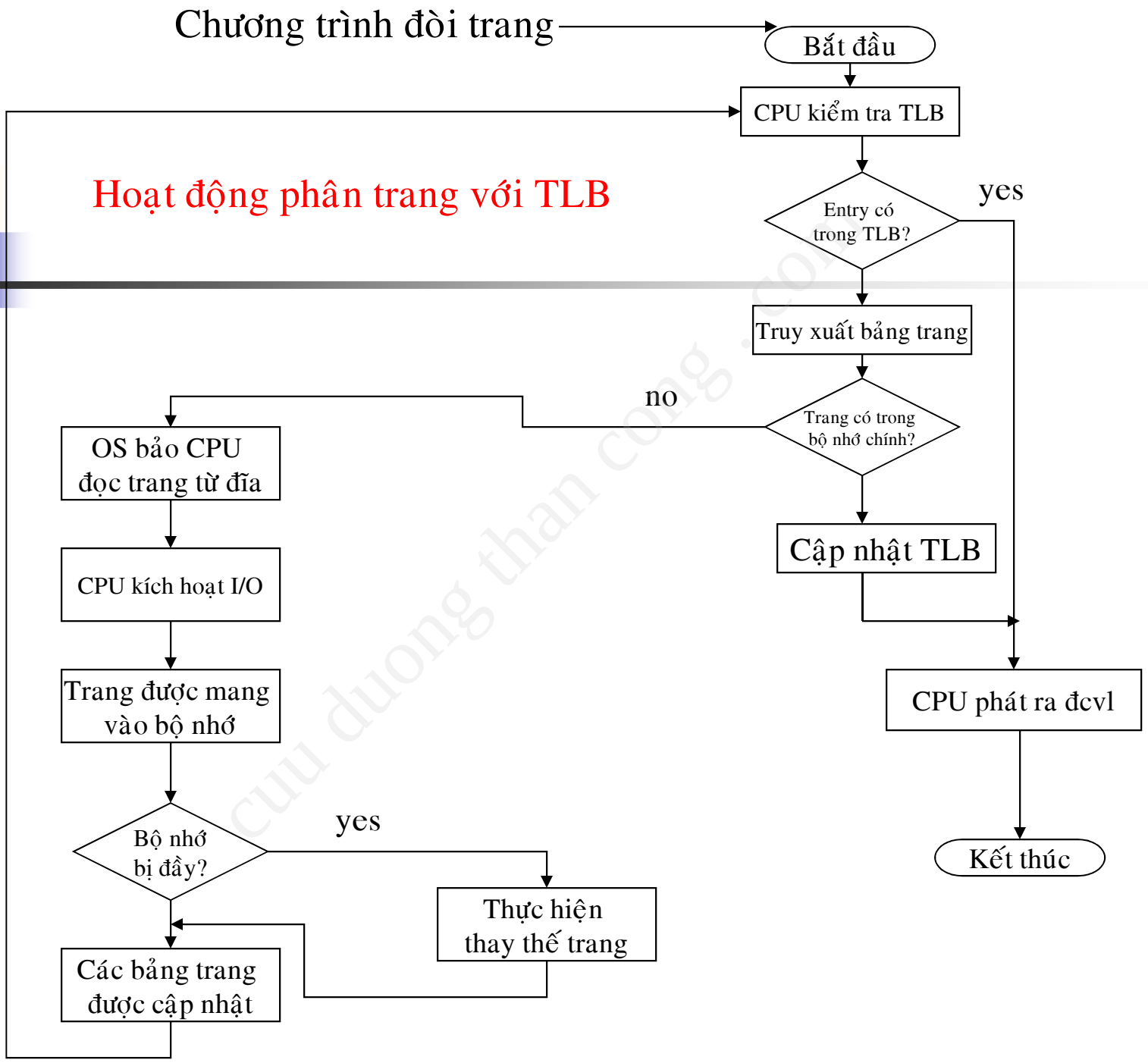
- Mỗi một tham chiếu bộ nhớ ảo sẽ gây ra hai truy xuất bộ nhớ vật lý:
  - Nạp mục thích hợp từ bảng trang
  - Nạp dữ liệu cần thiết

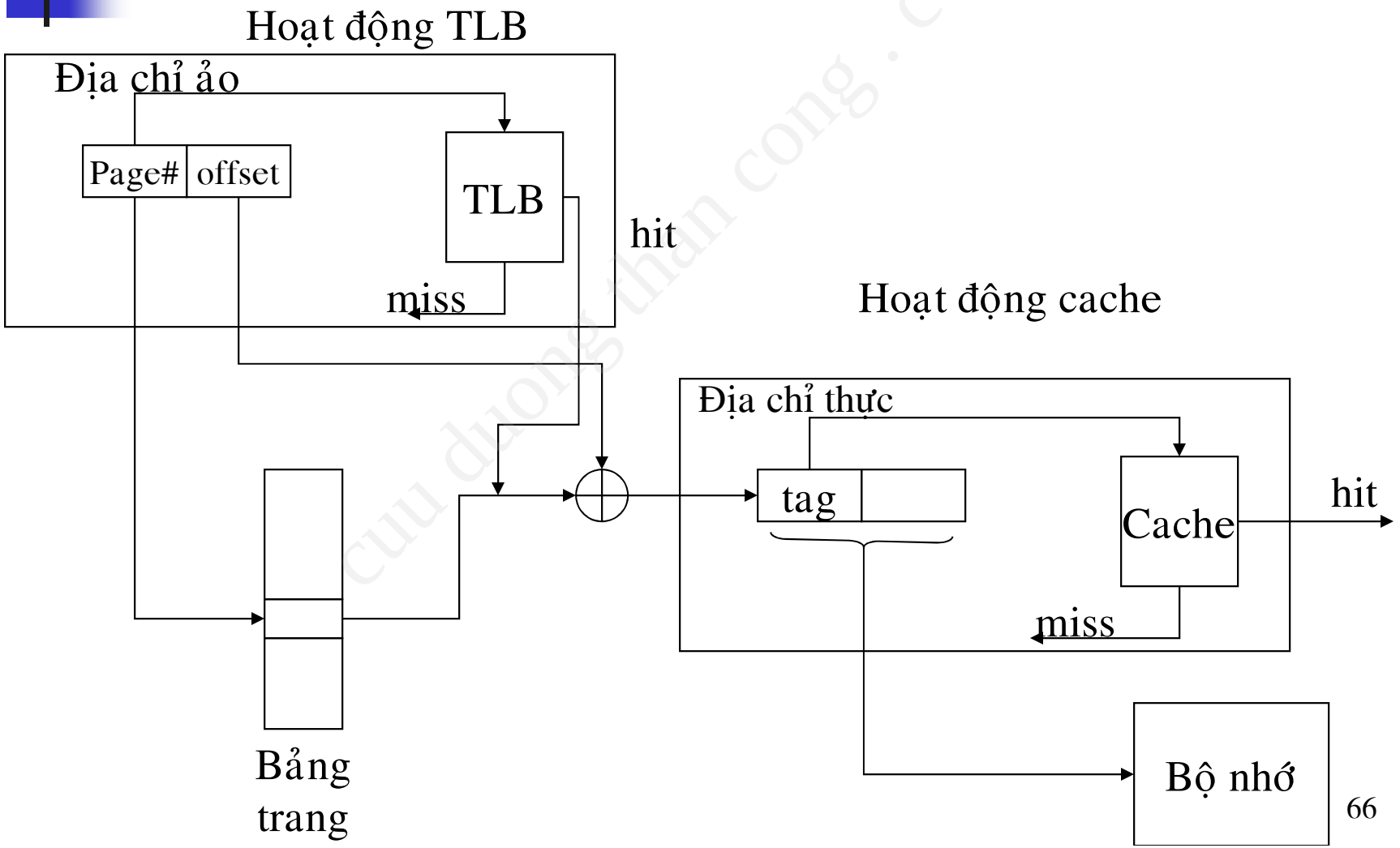
  =>gấp đôi thời gian truy xuất
- Dùng cache đặc biệt cho các entry của bảng, gọi là bộ đệm TLB. Chứa các entry được dùng trong thời gian gần nhất.

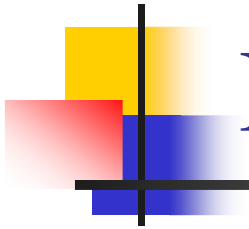
Bộ đệm TLB (Translation Lookaside Buffer)...

- Cơ cấu bộ nhớ ảo vẫn phải tương tác với cache hệ thống.
  - Mỗi khi một địa chỉ ảo phát ra, trước hết hệ thống bộ nhớ sẽ xem trong TLB có entry tương ứng không nếu có địa chỉ thực được tạo ra, nếu không sẽ truy xuất entry từ bảng trang. Khi địa chỉ thực phát ra sẽ truy xuất nội dung theo trình tự có tương tác với cache hệ thống như đã trình bày phần trước.
  - Tham chiếu entry có thể trong TLB, bộ nhớ chính hay trên đĩa.
  - Tham chiếu từ nhớ có thể trong cache, trong bộ nhớ chính hay trên đĩa. Nếu từ nhớ được lấy từ đĩa thì trang chứa nó sẽ được nạp vào bộ nhớ và khối nhớ chứa trang sẽ được nạp vào cache.

Hoạt động phân trang với TLB

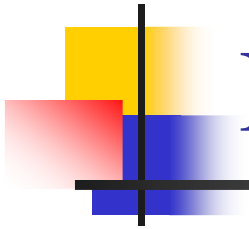






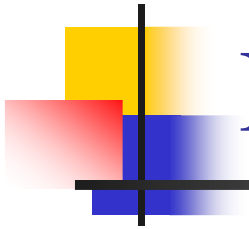
Phân đoạn (segmentation)

- Một giải pháp khác nhằm chia nhỏ bộ nhớ, được gọi là phân đoạn.
- Người lập trình không thể nhìn thấy và dùng sự phân trang, mục đích của phân trang là cung cấp một không gian địa chỉ lớn hơn thực tế.
- Người lập trình có thể nhận thấy được sự phân đoạn, nó cung cấp một phương tiện để tổ chức chương trình và dữ liệu, cũng là phương tiện liên quan đến các thuộc tính bảo vệ đối với chương trình và dữ liệu.
- Phân đoạn cho phép người lập trình xem bộ nhớ gồm nhiều không gian địa chỉ, gọi là segment. Segment có kích thước thay đổi linh hoạt.



Phân đoạn (segmentation)

- Thông thường người lập trình hay OS sẽ gán cho các chương trình và dữ liệu một số segment khác nhau. Số lượng segment sẽ thay đổi tuỳ vào chương trình dữ liệu
- Mỗi segment có thể được gán các quyền truy xuất và sử dụng.
- Các tham chiếu bộ nhớ bao gồm một dạng địa chỉ mới (chỉ số segment+offset)



Phân đoạn (segmentation)

- Phân đoạn có một số các ưu điểm sau:
  - Đơn giản trong việc kiểm soát khi có sự gia tăng cấu trúc dữ liệu. Cấu trúc dữ liệu sẽ được gán vào một segment và OS sẽ co giãn segment này khi cần.
  - Cho phép thay đổi và biên dịch lại các chương trình một cách độc lập (Setup)
  - Dễ dàng cho vay
  - Dễ dàng bảo vệ

Quản lý bộ nhớ trong Pentium

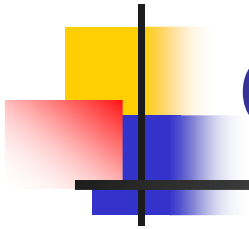
- Không gian địa chỉ: Phần cứng hỗ trợ cả Paging và Segmentation, có thể bị cấm để cho phép người dùng chọn trong số bốn kịch bản bộ nhớ sau:
  - Cấm cả hai: cho các ứng dụng điều khiển chất lượng cao.
  - Cấm segmentation: bộ nhớ được xem như một không gian địa chỉ tuyến tính phân trang. Bảo vệ và quản lý được thực hiện qua phân trang.
  - Cấm paging: bộ nhớ được xem như không gian địa chỉ luận lý. Cố gắng bảo vệ mức byte. Đảm bảo rằng bản thông dịch (segment table) ở trên chip khi segment đang trong bộ nhớ.=> thời gian truy xuất có thể dự báo được.
  - Cho phép cả hai: seg định nghĩa các phần bộ nhớ luận lý nhằm điều khiển truy xuất và paging để quản lý sự phân phối bộ nhớ bên trong các phần

Quản lý bộ nhớ trong Pentium...

- Segmentation: Mỗi địa chỉ ảo gồm 16 bit segment và 32 bit offset. 2 bit phần segment liên quan đến cơ cấu bảo vệ, 14 bit còn lại chỉ ra segment cụ thể→ cấm phân đoạn thì bộ nhớ ảo là 4Gb. Cho phân đoạn thì bộ nhớ là 64Tb.
  - Không gian địa chỉ ảo được chia thành 2 phần: toàn cục (chia sẻ cho tất cả các process) và cục bộ
  - Có hai dạng bảo vệ: privileged level và access attribute. Có bốn mức privilege 0 (cao nhất) đến 3 (thấp nhất). OS sẽ quyết định cách dùng các mức này.
  - 16 bit segment trong phần địa chỉ ảo gồm có bốn phần:
    - Tabe Indicator: chỉ ra nơi chứa bảng segment toàn cục hay cục bộ
    - Segment number: chỉ số của segment trong bảng
    - Requested Privileged Level: mức đặc quyền cần cho truy xuất này

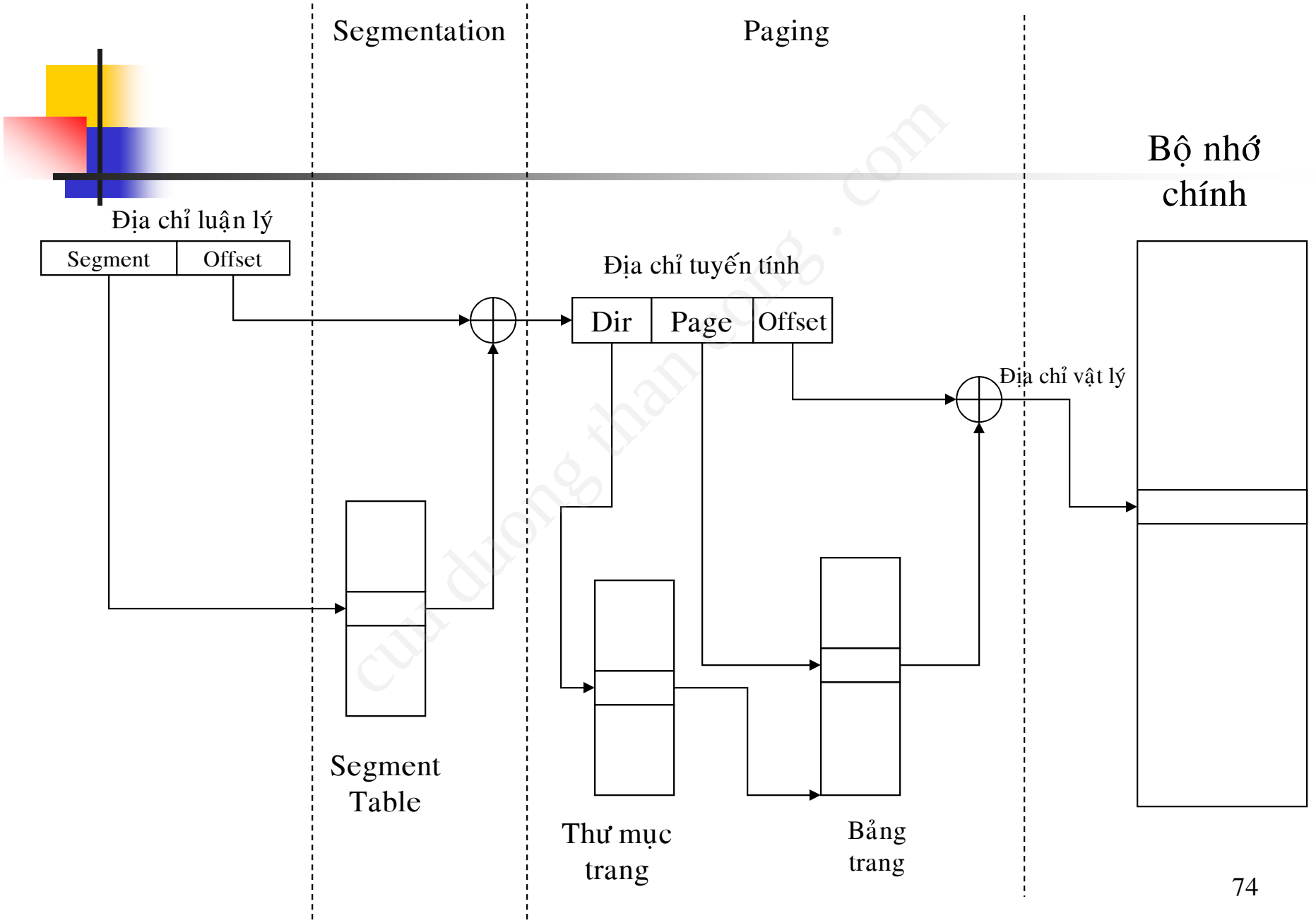
Quản lý bộ nhớ trong Pentium...

- Paging: khi dùng segmentation các địa chỉ được dùng là địa chỉ ảo và được đổi sang địa chỉ tuyến tính (32 bit). Khi không dùng segmentation thì địa chỉ tuyến tính được dùng. Trong cả hai trường hợp đều phải đổi sang 32 bit địa chỉ thực.
- Cơ cấu phân trang của Pentium là hoạt động dò bảng hai mức. Mức đầu là thư mục trang chứa 1024 mục. Chia 4Gb thành các nhóm 1024 nhóm trang có kích thước 4Mb. Mỗi nhóm ứng với 1 bảng trang có 1024 mục. Quản lý bộ nhớ có thể chọn dùng 1 thư mục trang cho tất cả process hoặc một cho mỗi process hoặc dạng kết hợp. Thư mục trang của tác vụ hiện hành luôn ở trong bộ nhớ. Mức hai là bảng trang, Pentium cũng dùng TLB, mỗi TLB có thể chứa 32 mục. Mỗi khi thư mục trang thay đổi, bộ đệm này bị xóa.
- Pentium cho phép hai kích thước trang: 4Kbyte hay 4Mbyte



Quản lý bộ nhớ trong Pentium...

- Khi trang 4Mbyte được dùng chỉ có một mức dò cho các trang.
- Dùng 4Mbyte page giảm đi nhu cầu lưu trữ bộ nhớ quản lý. Nếu dùng 4Kbyte page thì cần khoảng 4Mbyte để chứa bảng trang. Với 4Mbyte page thì chỉ cần 4Kbyte là đủ cho việc quản lý bộ nhớ trang.



HẾT