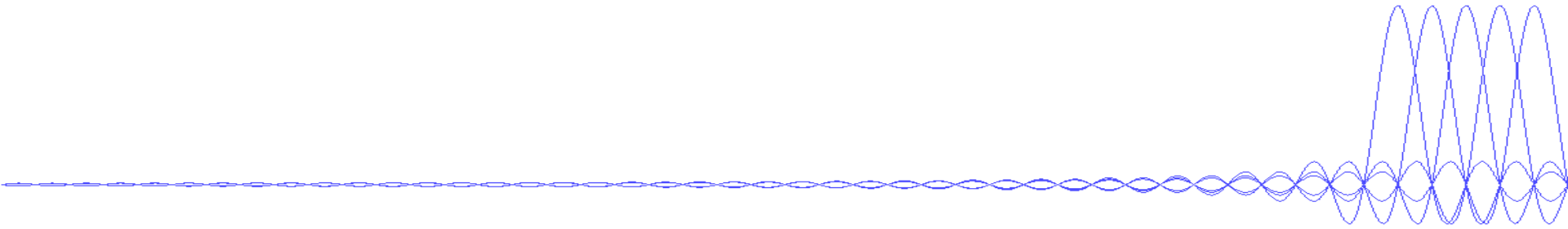
COMPUTER ENGINEERING

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

HỆ ĐIỀU HÀNH
Chương 7 – Quản lý bộ nhớ (1)

14/03/2017



COMPUTER ENGINEERING

Câu hỏi ôn tập chương 6

- Nêu điều kiện để thực hiện giải thuật Banker?
- Nêu các bước của giải thuật Banker?
- Nêu các bước của giải thuật yêu cầu tài nguyên?
- Nêu các bước giải thuật phát hiện deadlock?
- Khi deadlock xảy ra, hệ điều hành làm gì để phục hồi?
- Dựa trên yếu tố nào để chấm dứt quá trình bị deadlock??

COMPUTER ENGINEERING

Câu hỏi ôn tập chương 6 (tt)

- Cho 1 hệ thống có 4 tiến trình P1 đến P4 và 3 loại tài nguyên R1 (3), R2 (2) R3 (2). P1 giữ 1 R1 và yêu cầu 1 R2; P2 giữ 2 R2 và yêu cầu 1 R1 và 1 R3; P3 giữ 1 R1 và yêu cầu 1 R2; P4 giữ 2 R3 và yêu cầu 1 R1
  - Vẽ đồ thị tài nguyên cho hệ thống này?
  - Deadlock?
  - Chuỗi an toàn? (nếu có)

Câu hỏi ôn tập chương 6 (tt)

- Tìm Need?
- Hệ thống có an toàn không?
- Nếu P1 yêu cầu (0,4,2,0) thì có thể cấp phát cho nó ngay không?

	Allocation	*Max*	*Available*
	A B C D	*A B C D*	*A B C D*
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 7 5 0	
P_2	1 3 5 4	2 3 5 6	
P_3	0 6 3 2	0 6 5 2	
P_4	0 0 1 4	0 6 5 6	

COMPUTER ENGINEERING

Mục tiêu chương 7-1

- Hiểu được các khái niệm cơ sở về bộ nhớ
- Hiểu được các kiểu địa chỉ nhớ và cách chuyển đổi giữa các kiểu này
- Hiểu được các cơ chế và mô hình quản lý bộ nhớ

COMPUTER ENGINEERING

Nội dung chương 7-1

- Khái niệm cơ sở
- Các kiểu địa chỉ nhớ
- Chuyển đổi địa chỉ nhớ
- Overlay và swapping
- Mô hình quản lý bộ nhớ

COMPUTER ENGINEERING

Khái niệm cơ sở

- Chương trình phải được mang vào trong bộ nhớ và đặt nó trong một tiến trình để được xử lý
- Input Queue – Một tập hợp của những tiến trình trên đĩa mà đang chờ để được mang vào trong bộ nhớ để thực thi.
- User programs trải qua nhiều bước trước khi được xử lý.

COMPUTER ENGINEERING

Khái niệm cơ sở (tt)

- Quản lý bộ nhớ là công việc của hệ điều hành với sự hỗ trợ của phần cứng nhằm phân phối, sắp xếp các process trong bộ nhớ sao cho hiệu quả.
- Mục tiêu cần đạt được là nạp càng nhiều process vào bộ nhớ càng tốt (gia tăng mức độ đa chương)
- Trong hầu hết các hệ thống, kernel sẽ chiếm một phần cố định của bộ nhớ; phần còn lại phân phối cho các process.

COMPUTER ENGINEERING

Khái niệm cơ sở (tt)

- Các yêu cầu đối với việc quản lý bộ nhớ
  - Cấp phát bộ nhớ cho các process
  - Tái định vị (relocation): khi swapping,…
  - Bảo vệ: phải kiểm tra truy xuất bộ nhớ có hợp lệ không
  - Chia sẻ: cho phép các process chia sẻ vùng nhớ chung
  - Kết gán địa chỉ nhớ luận lý của user vào địa chỉ thực

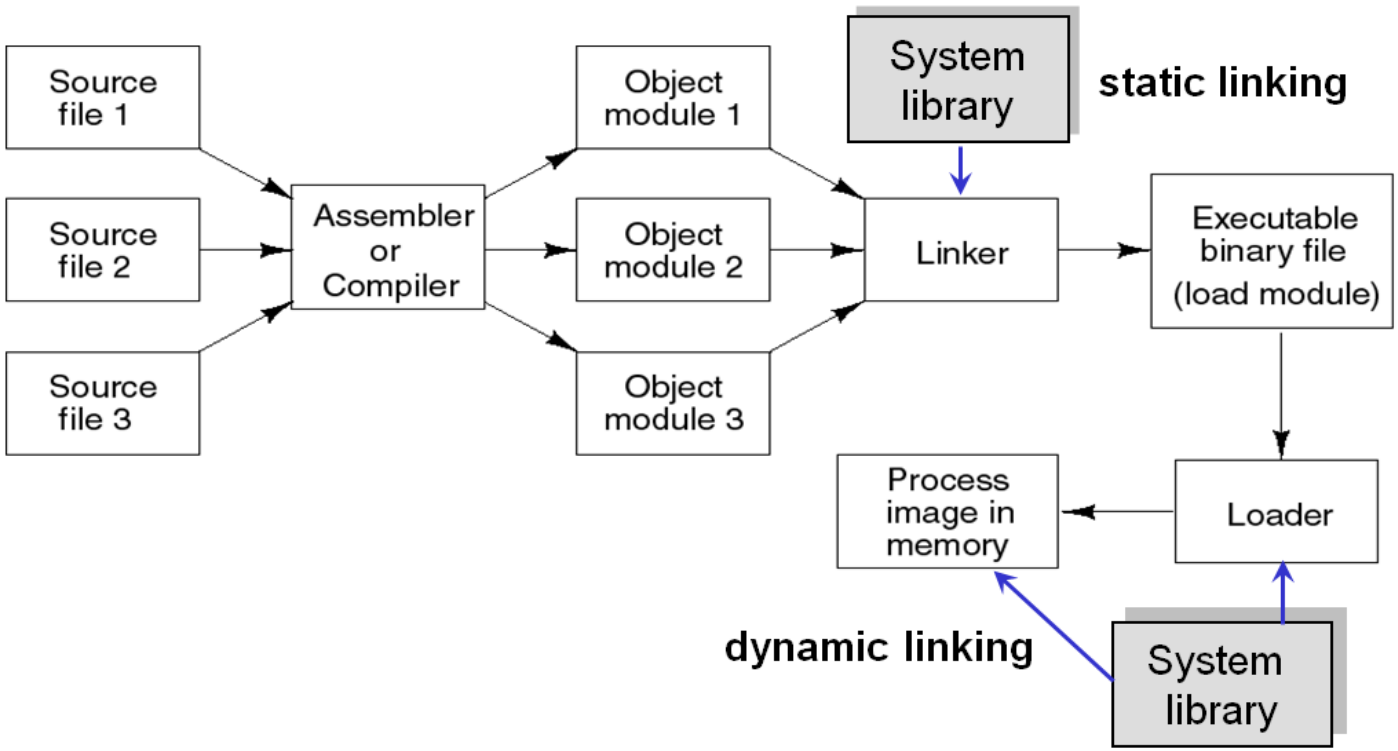
COMPUTER ENGINEERING

Cấu trúc dữ liệu cho giải thuật Banker

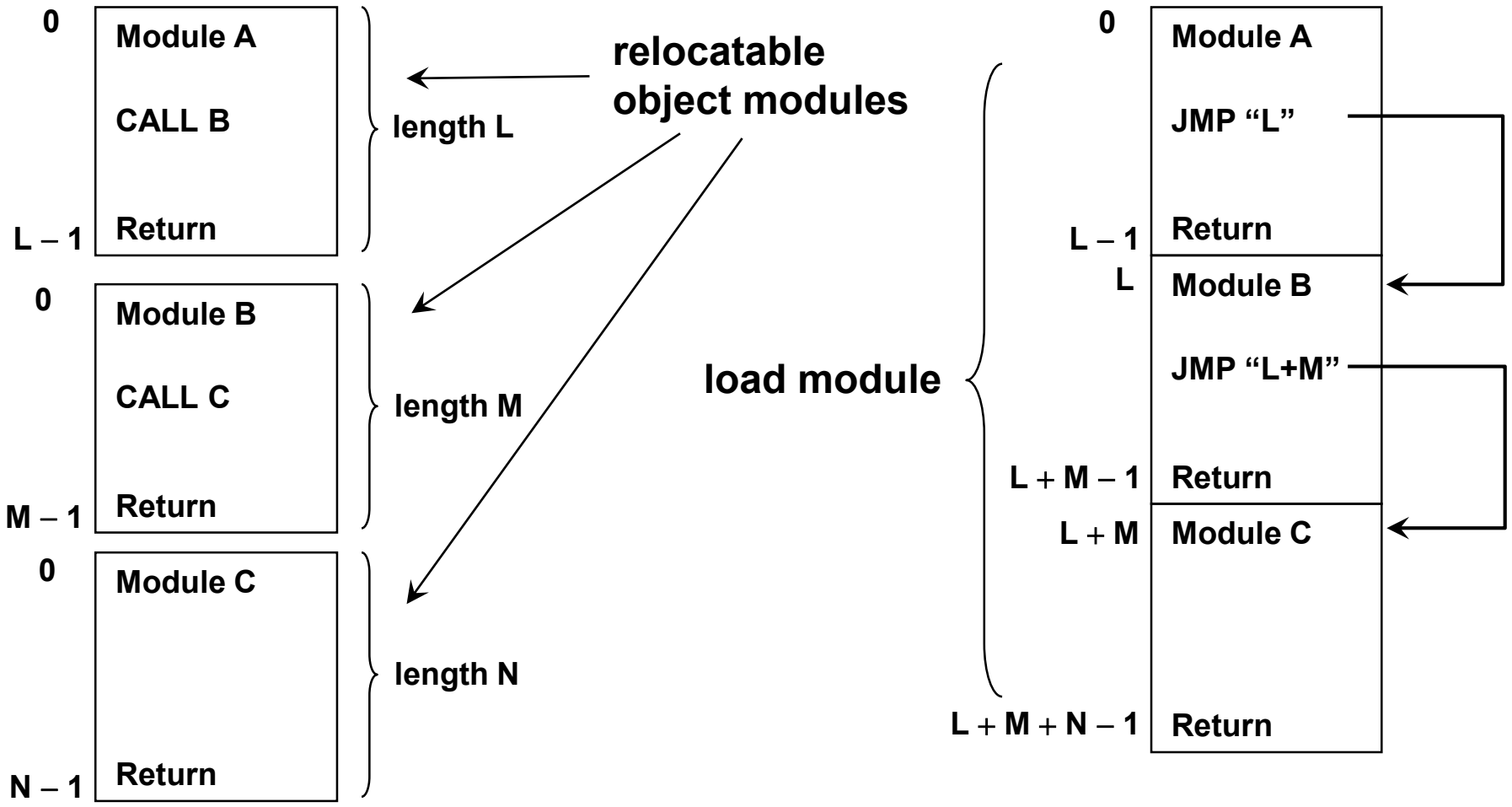
- Địa chỉ vật lý (physical address) (địa chỉ thực) là một vị trí thực trong bộ nhớ chính
- Địa chỉ luận lý (logical address) là một vị trí nhớ được diễn tả trong một chương trình (còn gọi là địa chỉ ảo virtual address).
  - Các trình biên dịch (compiler) tạo ra mã lệnh chương trình mà trong đó mọi tham chiếu bộ nhớ đều là địa chỉ luận lý
  - Địa chỉ tương đối (relative address) (địa chỉ khả tái định vị, relocatable address) là một kiểu địa chỉ luận lý trong đó các địa chỉ được biểu diễn tương đối so với một vị trí xác định nào đó trong chương trình.
    - Ví dụ: 12 byte so với vị trí bắt đầu chương trình,…
  - Địa chỉ tuyệt đối (absolute address): địa chỉ tương đương với địa chỉ thực.

Nạp chương trình vào bộ nhớ

- Bộ linker: kết hợp các object module thành một file nhị phân khả thực thi gọi là load module.
- Bộ loader: nạp load module vào bộ nhớ chính

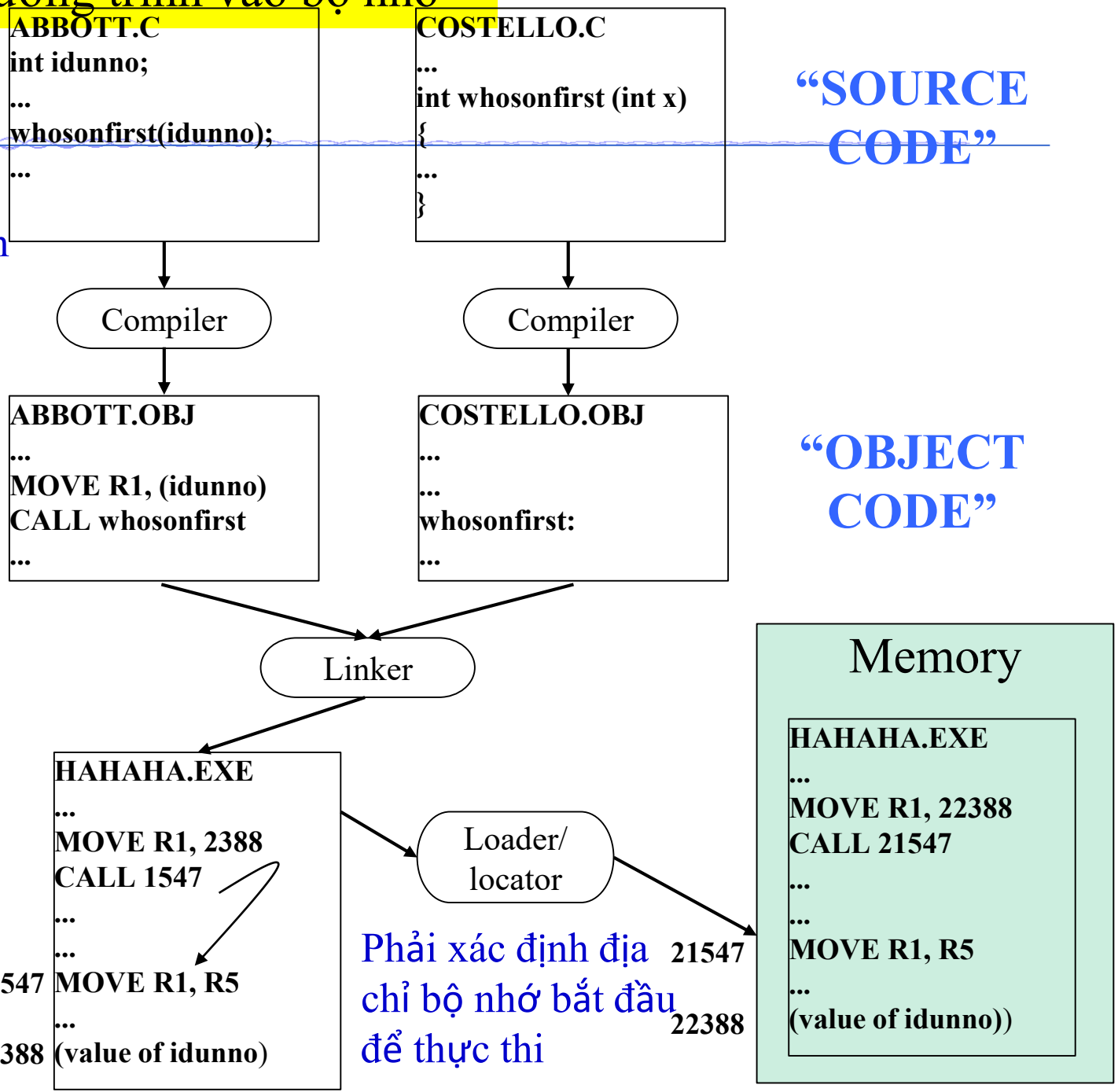


Cơ chế thực hiện linking



Các bước nạp chương trình vào bộ nhớ

COMPUTER ENGINEERING



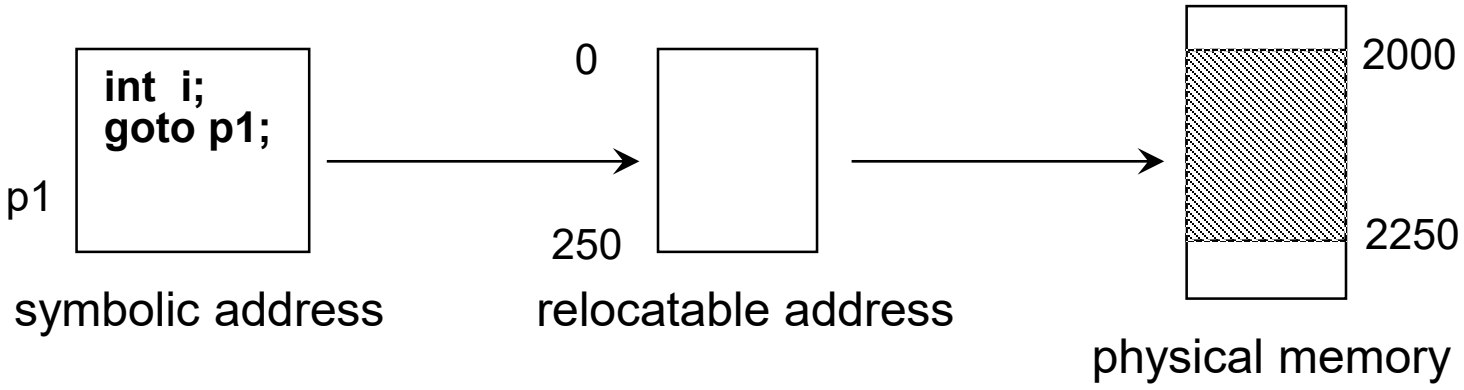
Khi mỗi file được biên dịch, các địa chỉ chưa biết, vì thế các cờ được dùng để đánh dấu

Trình linker kết nối các files, vì thế nó có thể thay thế các chỗ đánh dấu với địa chỉ thật

Phải xác định địa chỉ bộ nhớ bắt đầu để thực thi

Chuyển đổi địa chỉ

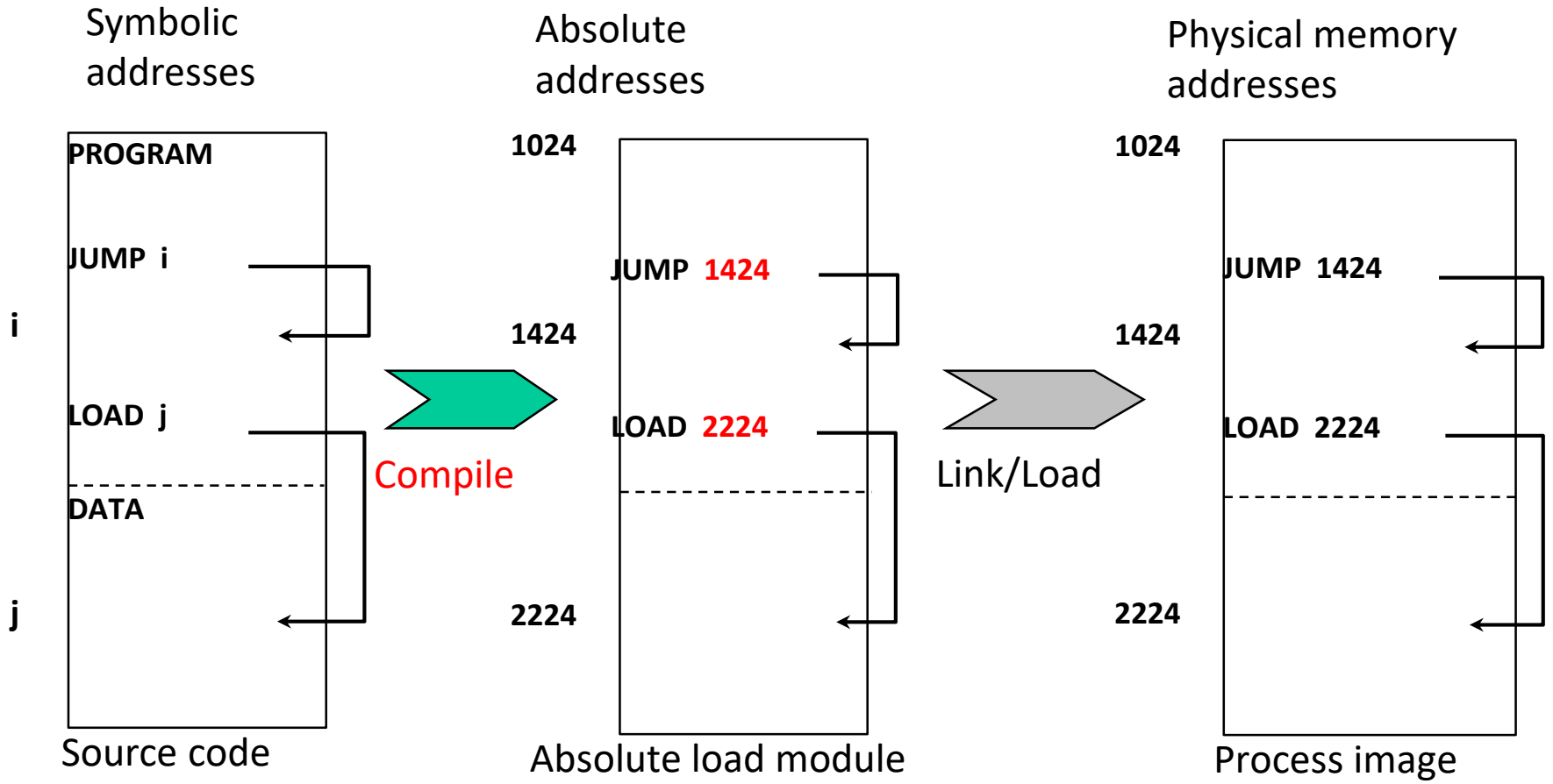
- Chuyển đổi địa chỉ: quá trình ánh xạ một địa chỉ từ không gian địa chỉ này sang không gian địa chỉ khác.
- Biểu diễn địa chỉ nhớ
  - Trong source code: symbolic (các biến, hằng, pointer,…)
  - Trong thời điểm biên dịch: thường là địa chỉ khả tái định vị
    - Ví dụ: a ở vị trí 12 byte so với vị trí bắt đầu module
  - Thời điểm liking/loading: có thể là địa chỉ thực.
    - Ví dụ: dữ liệu nằm tại địa chỉ bộ nhớ thực 2030



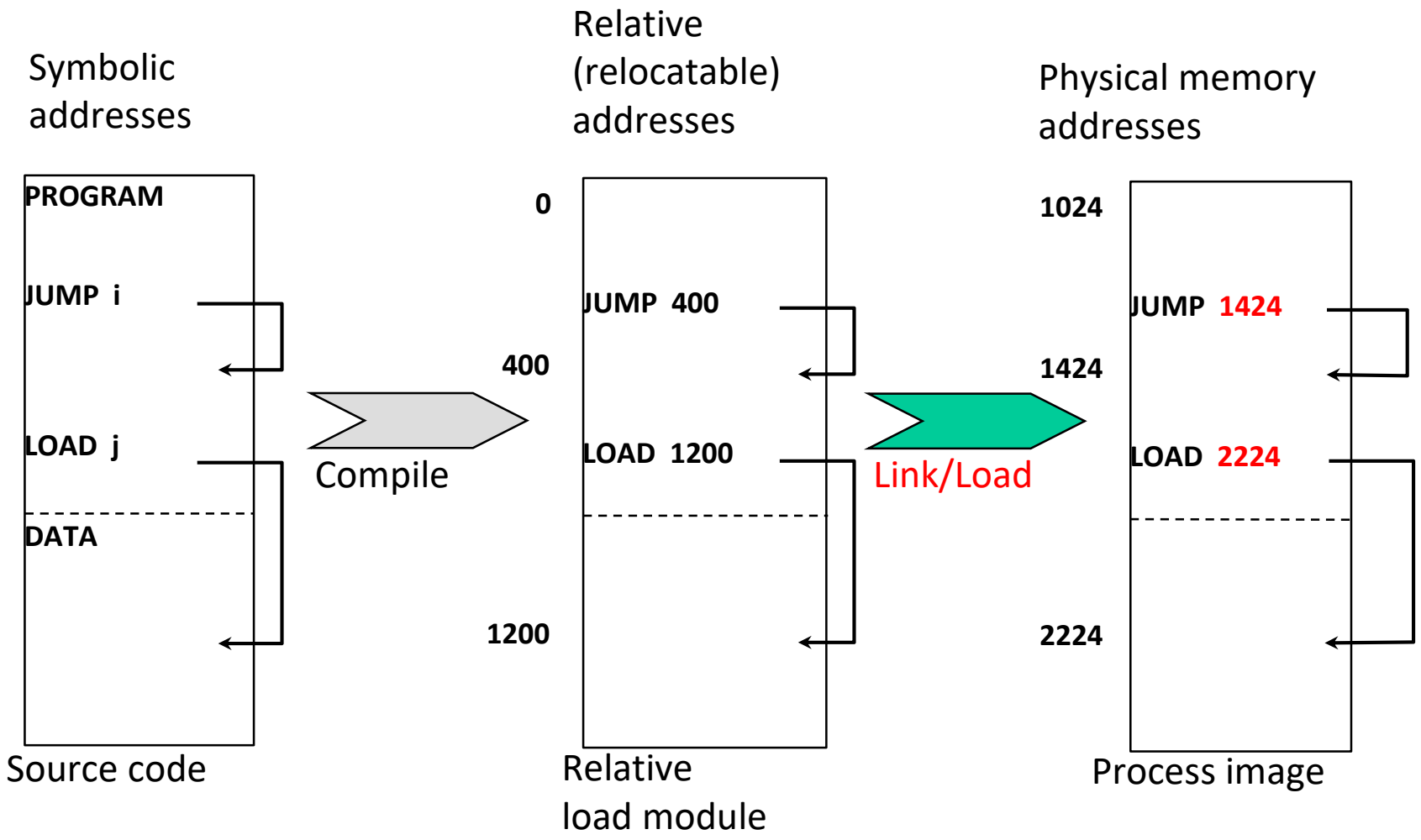
Chuyển đổi địa chỉ (tt)

- Địa chỉ lệnh và dữ liệu được chuyển đổi thành địa chỉ thực có thể xảy ra tại ba thời điểm khác nhau.
  - Compile time: nếu biết trước địa chỉ bộ nhớ của chương trình thì có thể kết gán địa chỉ tuyệt đối lúc biên dịch
    - Ví dụ: chương trình .COM của MS-DOS
    - Khuyết điểm: phải biên dịch lại nếu thay đổi địa chỉ nạp chương trình
  - Load time: vào thời điểm loading, loader phải chuyển đổi địa chỉ khả tái định vị thành địa chỉ thực dựa trên một địa chỉ nền
    - Địa chỉ thực được tính toán vào thời điểm nạp chương trình => phải tiến hành reload nếu địa chỉ nền thay đổi

Sinh địa chỉ tuyệt đối vào thời điểm dịch

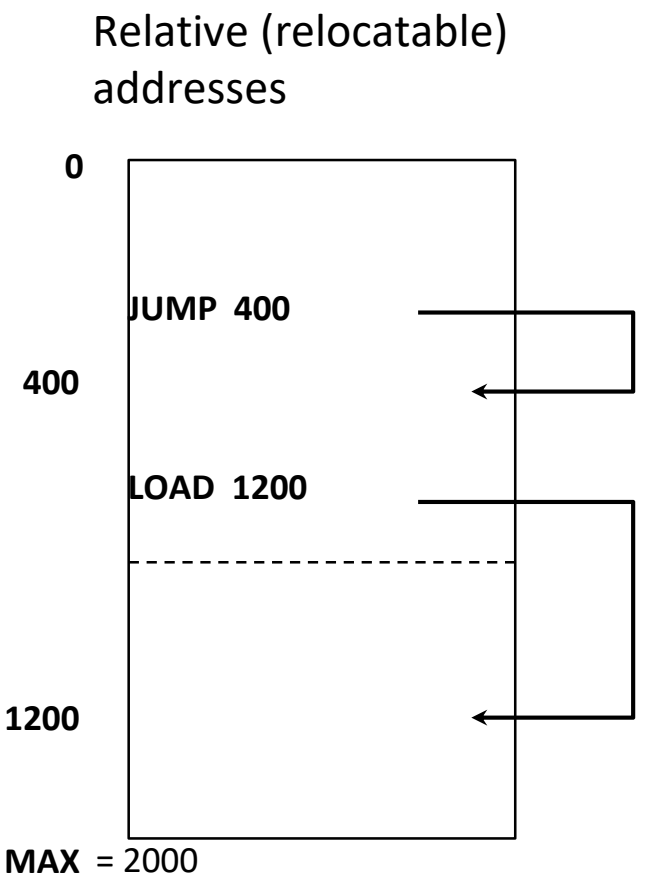


Sinh địa chỉ tuyệt đối vào thời điểm nạp



Chuyển đổi địa chỉ (tt)

- Excution time: khi trong quá trình thực thi, process có thể được di chuyển từ segment này sang segment khác trong bộ nhớ thì quá trình chuyển đổi địa chỉ được trì hoãn đến thời điểm thực thi
  - Cần sự hỗ trợ của phần cứng cho việc ánh xạ địa chỉ
    - Ví dụ: Trường hợp địa chỉ luận lý là relocatable thì có thể dùng thanh ghi base và limit,..
  - Sử dụng trong đa số các OS đa dụng trong đó có các cơ chế swapping, paging, segmentation



COMPUTER ENGINEERING

Dynamic linking

- Quá trình link đến một module ngoài (external module) được thực hiện sau khi đã tạo xong load module (i.e. file có thể thực thi, executable)
  - Ví dụ trong Windows: module ngoài là các file .DLL còn trong Unix, các module ngoài là các file .so (shared library)
- Load module chứa các stub tham chiếu (refer) đến routine của external module.
  - Lúc thực thi, khi stub được thực thi lần đầu (do process gọi routine lần đầu), stub nạp routine vào bộ nhớ, tự thay thế bằng địa chỉ của routine và routine được thực thi.
  - Các lần gọi routine sau sẽ xảy ra bình thường
- Stub cần sự hỗ trợ của OS (như kiểm tra xem routine đã được nạp vào bộ nhớ chưa).

COMPUTER ENGINEERING

Ưu điểm của dynamic linking

- Thông thường, external module là một thư viện cung cấp các tiện ích của OS. Các chương trình thực thi có thể dùng các phiên bản khác nhau của external module mà không cần sửa đổi, biên dịch lại.
- Chia sẻ mã (code sharing): một external module chỉ cần nạp vào bộ nhớ một lần. Các process cần dùng external module này thì cùng chia sẻ đoạn mã của external module ⇒ tiết kiệm không gian nhớ và đĩa.
- Phương pháp dynamic linking cần sự hỗ trợ của OS trong việc kiểm tra xem một thủ tục nào đó có thể được chia sẻ giữa các process hay là phần mã của riêng một process (bởi vì chỉ có OS mới có quyền thực hiện việc kiểm tra này).

COMPUTER ENGINEERING

Dynamic loading

- Cơ chế: chỉ khi nào cần được gọi đến thì một thủ tục mới được nạp vào bộ nhớ chính ⇒ tăng độ hiệu dụng của bộ nhớ bởi vì các thủ tục không được gọi đến sẽ không chiếm chỗ trong bộ nhớ
- Rất hiệu quả trong trường hợp tồn tại khối lượng lớn mã chương trình có tần suất sử dụng thấp, không được sử dụng thường xuyên (ví dụ các thủ tục xử lý lỗi)
- Hỗ trợ từ hệ điều hành
  - Thông thường, user chịu trách nhiệm thiết kế và hiện thực các chương trình có dynamic loading.
  - Hệ điều hành chủ yếu cung cấp một số thủ tục thư viện hỗ trợ, tạo điều kiện dễ dàng hơn cho lập trình viên.

COMPUTER ENGINEERING

Cơ chế phủ lắp (overlay)

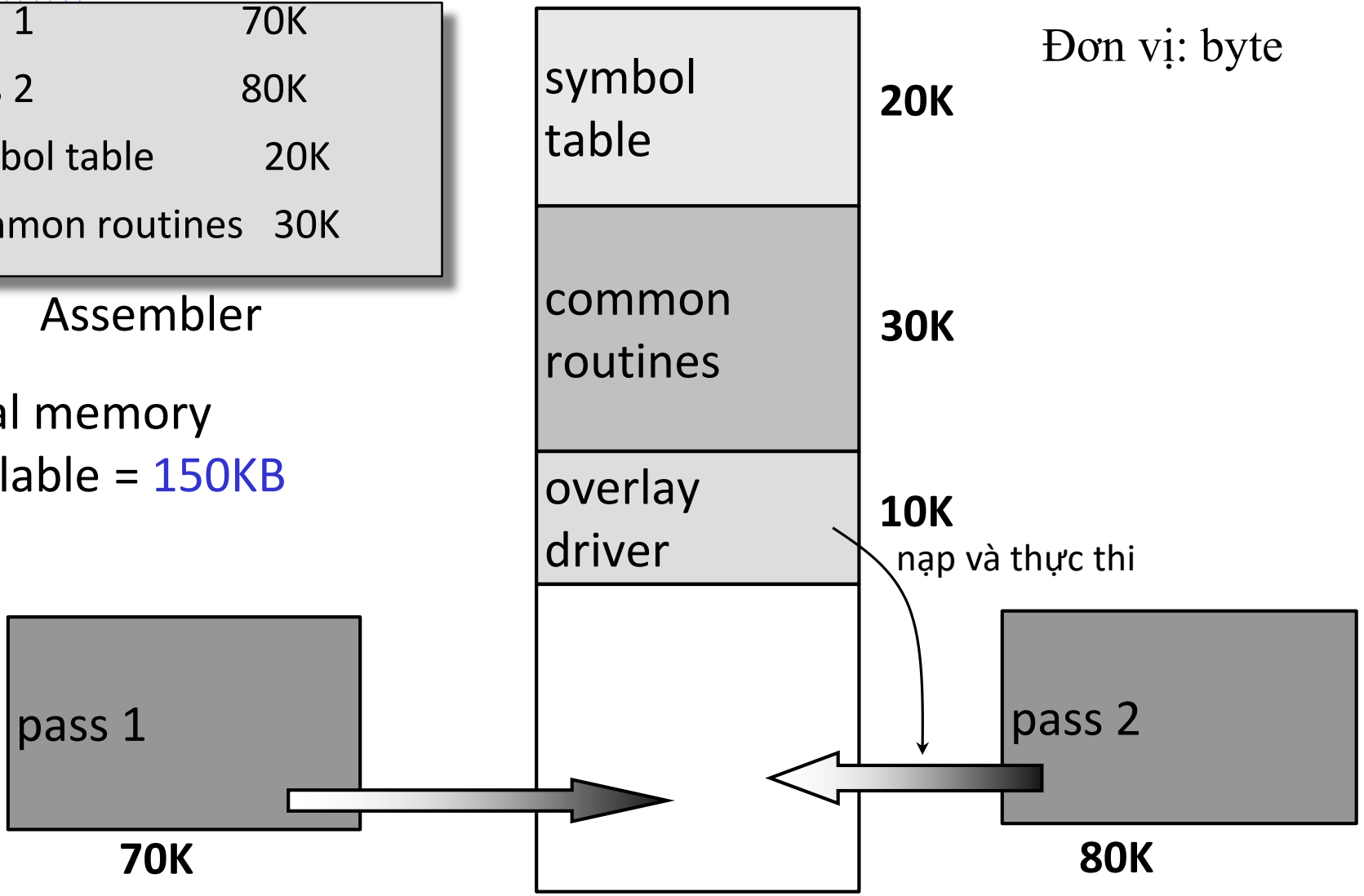
- Tại mỗi thời điểm, chỉ giữ lại trong bộ nhớ những lệnh hoặc dữ liệu cần thiết, giải phóng các lệnh/dữ liệu chưa hoặc không cần dùng đến.
- Cơ chế này rất hữu dụng khi kích thước một process lớn hơn không gian bộ nhớ cấp cho process đó.
- Cơ chế này được điều khiển bởi người sử dụng (thông qua sự hỗ trợ của các thư viện lập trình) chứ không cần sự hỗ trợ của hệ điều hành

Cơ chế phủ lắp (tt)

Pass 1	70K
Pass 2	80K
Symbol table	20K
Common routines	30K

Assembler

Total memory available = 150KB

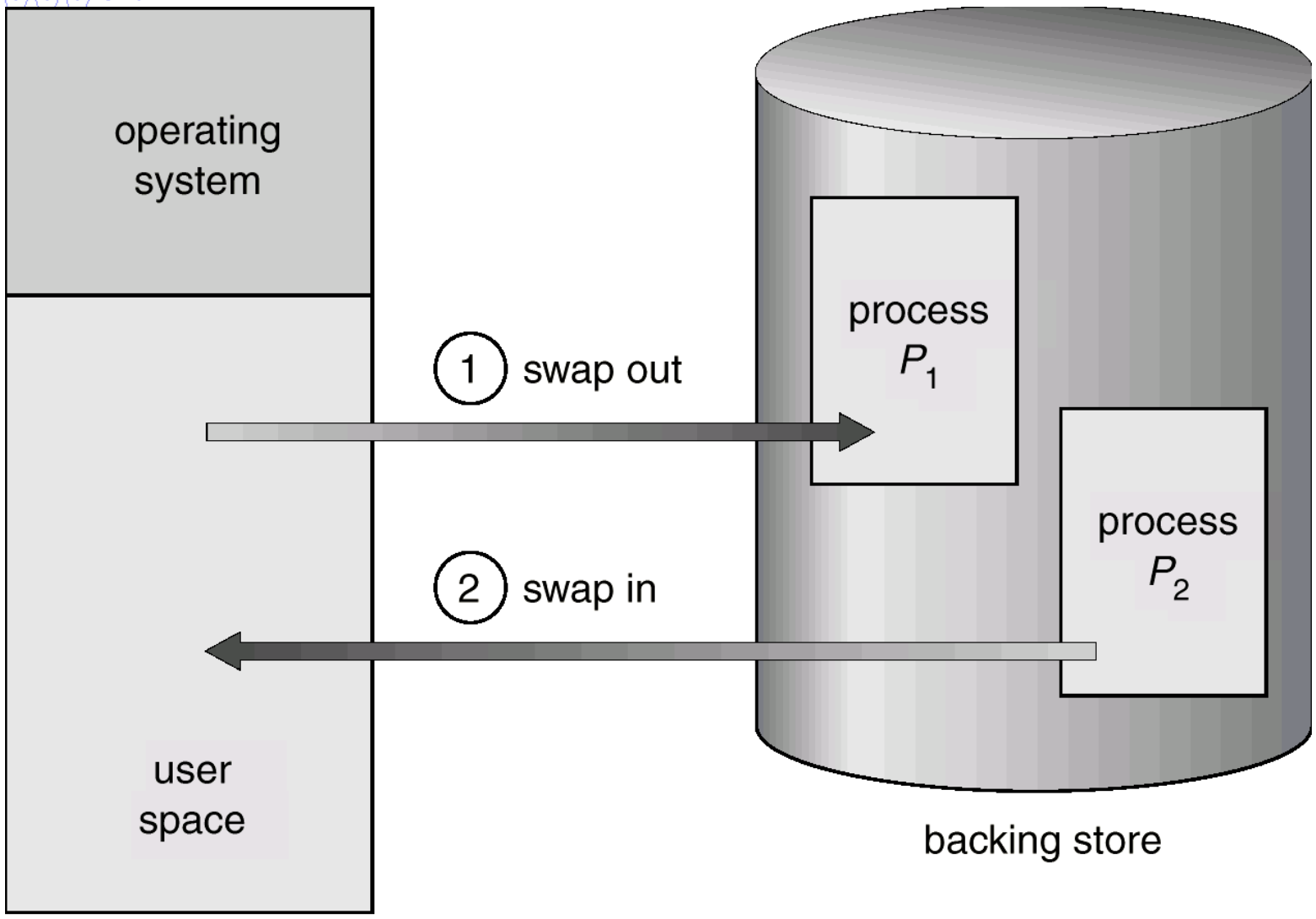


COMPUTER ENGINEERING

Cơ chế hoán vị (swapping)

- Một process có thể tạm thời bị swap ra khỏi bộ nhớ chính và lưu trên một hệ thống lưu trữ phụ. Sau đó, process có thể được nạp lại vào bộ nhớ để tiếp tục quá trình thực thi.
- Swapping policy: hai ví dụ
  - Round-robin: swap out P1 (vừa tiêu thụ hết quantum của nó), swap in P2 , thực thi P3 ,…
  - Roll out, roll in: dùng trong cơ chế định thời theo độ ưu tiên (priority-based scheduling)
    - Process có độ ưu tiên thấp hơn sẽ bị swap out nhường chỗ cho process có độ ưu tiên cao hơn mới đến được nạp vào bộ nhớ để thực thi
- Hiện nay, ít hệ thống sử dụng cơ chế swapping trên

Minh họa cơ chế hoán vị

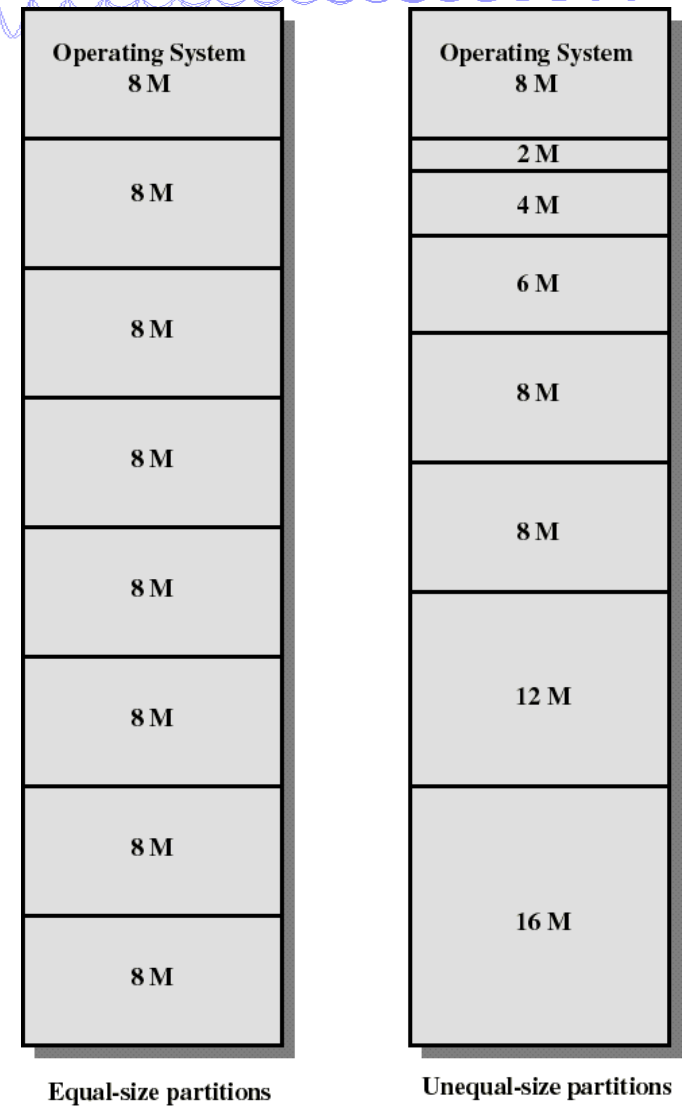


COMPUTER ENGINEERING

Mô hình quản lý bộ nhớ

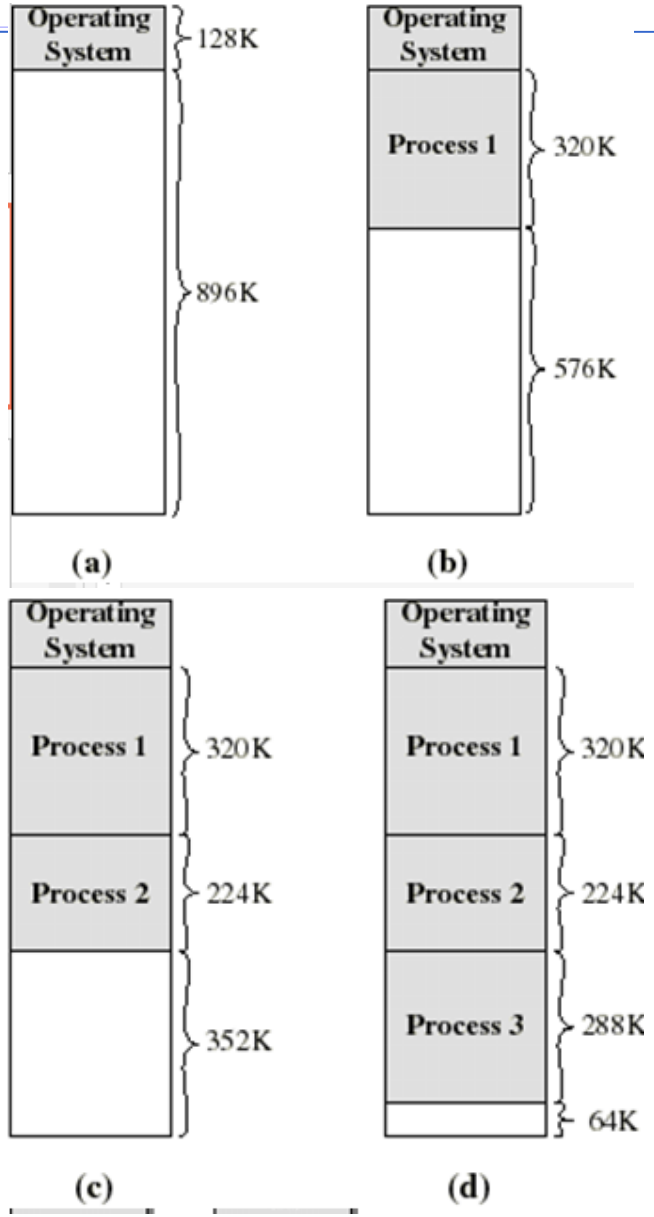
- Trong chương này, mô hình quản lý bộ nhớ là một mô hình đơn giản, không có bộ nhớ ảo.
- Một process phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi (ngoại trừ khi sử dụng cơ chế overlay).
- Các cơ chế quản lý bộ nhớ sau đây rất ít (hầu như không còn) được dùng trong các hệ thống hiện đại
  - Phân chia cố định (fixed partitioning)
  - Phân chia động (dynamic partitioning)
  - Phân trang đơn giản (simple paging)
  - Phân đoạn đơn giản (simple segmentation)

Mô hình quản lý bộ nhớ (tt)



Equal-size partitions

Unequal-size partitions

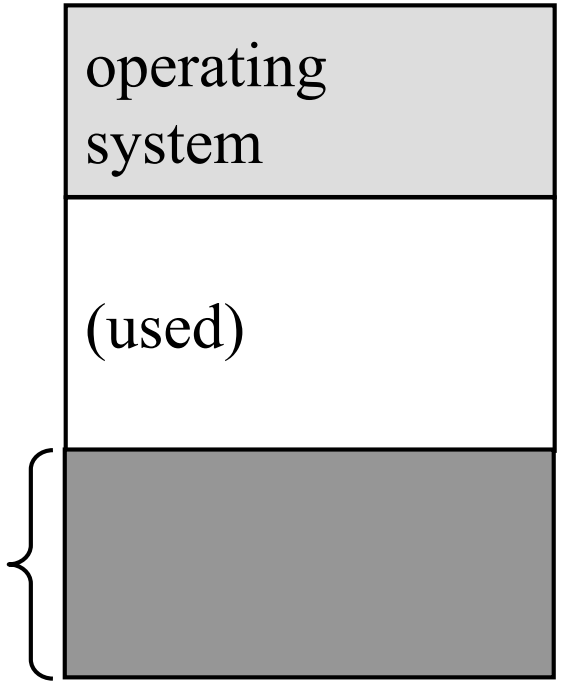


COMPUTER ENGINEERING

Phân mảnh (fragmentation)

- Phân mảnh ngoại (external fragmentation)
  - Kích thước không gian nhớ còn trống đủ để thỏa mãn một yêu cầu cấp phát, tuy nhiên không gian nhớ này không liên tục ⇒ có thể dùng cơ chế kết khối (compaction) để gom lại thành vùng nhớ liên tục.
- Phân mảnh nội (internal fragmentation)
  - Kích thước vùng nhớ được cấp phát có thể hơi lớn hơn vùng nhớ yêu cầu.
    - Ví dụ: cấp một khoảng trống 18,464 bytes cho một process yêu cầu 18,462 bytes.
  - Hiện tượng phân mảnh nội thường xảy ra khi bộ nhớ thực được chia thành các khối kích thước cố định (fixed-sized block) và các process được cấp phát theo đơn vị khối. Ví dụ: cơ chế phân trang (paging).

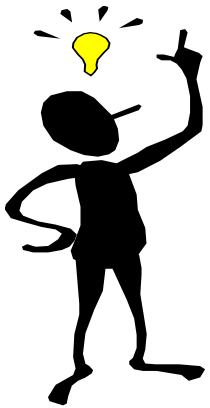
Phân mảnh nội



hole kích thước 18,464 bytes

Yêu cầu kế tiếp là 18,462 bytes !!!

Cần quản lý khoảng trống 2 bytes !?!

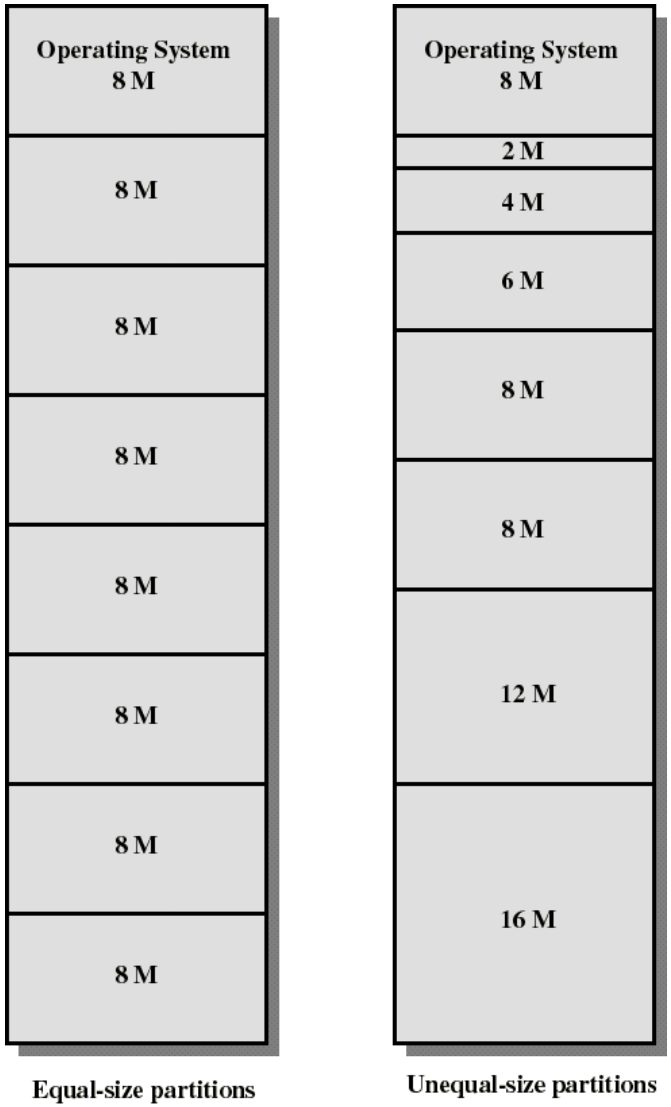


OS sẽ cấp phát hẳn khối18,464 bytes cho process
$\Rightarrow$ dư ra 2 bytes không dùng!

COMPUTER ENGINEERING

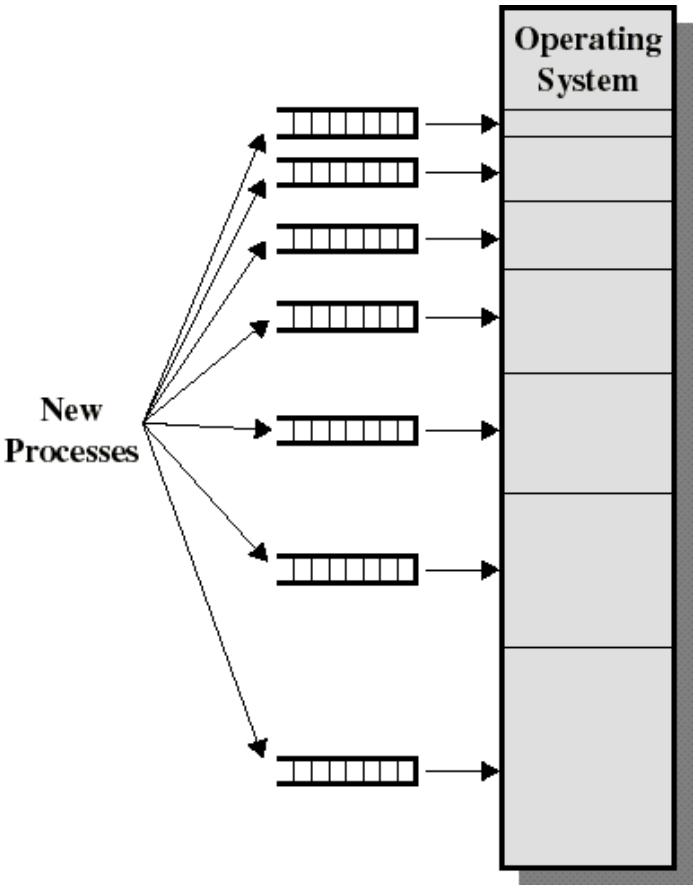
Fixed partitioning

- Khi khởi động hệ thống, bộ nhớ chính được chia thành nhiều phần rời nhau gọi là các partition có kích thước bằng nhau hoặc khác nhau
- Process nào có kích thước nhỏ hơn hoặc bằng kích thước partition thì có thể được nạp vào partition đó.
- Nếu chương trình có kích thước lớn hơn partition thì phải dùng cơ chế overlay.
- Nhận xét
  - Không hiệu quả do bị phân mảnh nội: một chương trình dù lớn hay nhỏ đều được cấp phát trọn một partition.



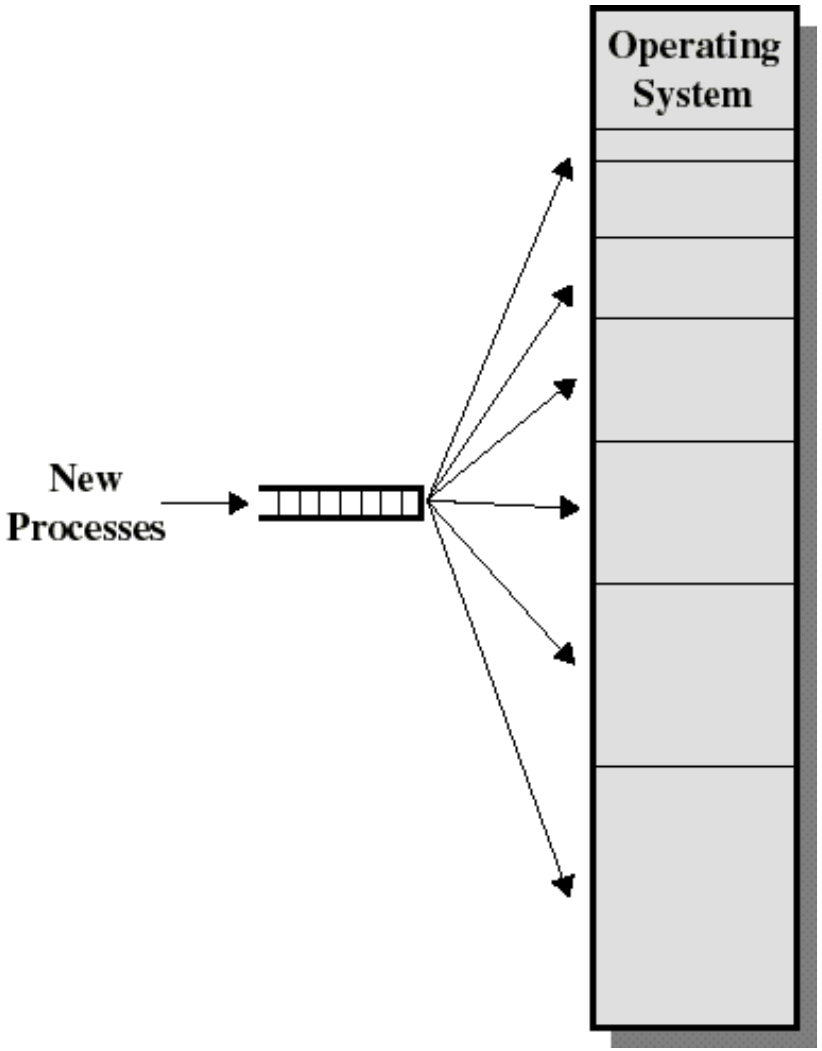
Chiến lược placement

- Partition có kích thước bằng nhau
  - Nếu còn partition trống ⇒ process mới sẽ được nạp vào partition đó
  - Nếu không còn partition trống, nhưng trong đó có process đang bị blocked ⇒ swap process đó ra bộ nhớ phụ nhường chỗ cho process mới.
- Partition có kích thước không bằng nhau: giải pháp 1
  - Gán mỗi process vào partition nhỏ nhất phù hợp với nó
  - Có hàng đợi cho mỗi partition
  - Giảm thiểu phân mảnh nội
  - Vấn đề: có thể có một số hàng đợi trống không (vì không có process với kích thước tương ứng) và hàng đợi dày đặc



Chiến lược placement (tt)

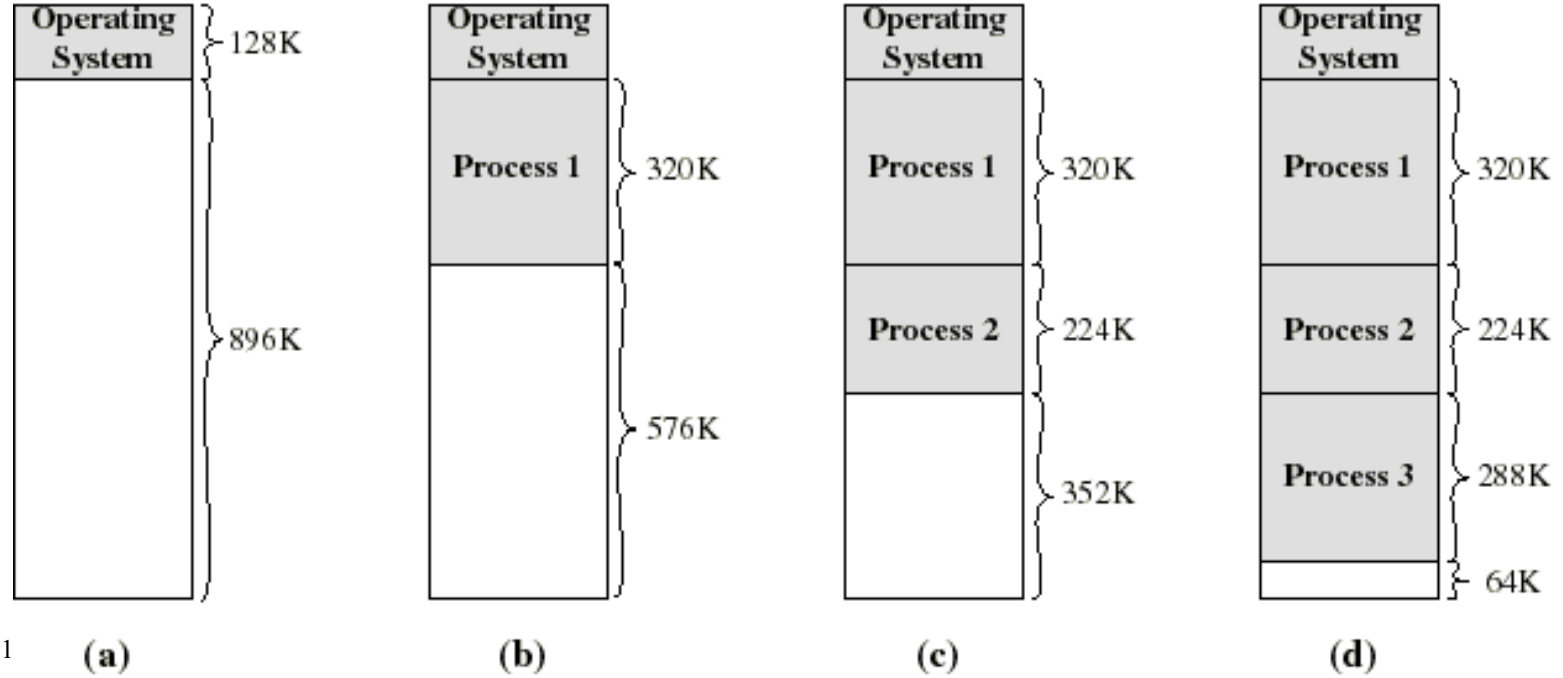
- Partition có kích thước không bằng nhau: giải pháp 2
  - Chỉ có một hàng đợi chung cho mọi partition
  - Khi cần nạp một process vào bộ nhớ chính ⇒ chọn partition nhỏ nhất còn trống



COMPUTER ENGINEERING

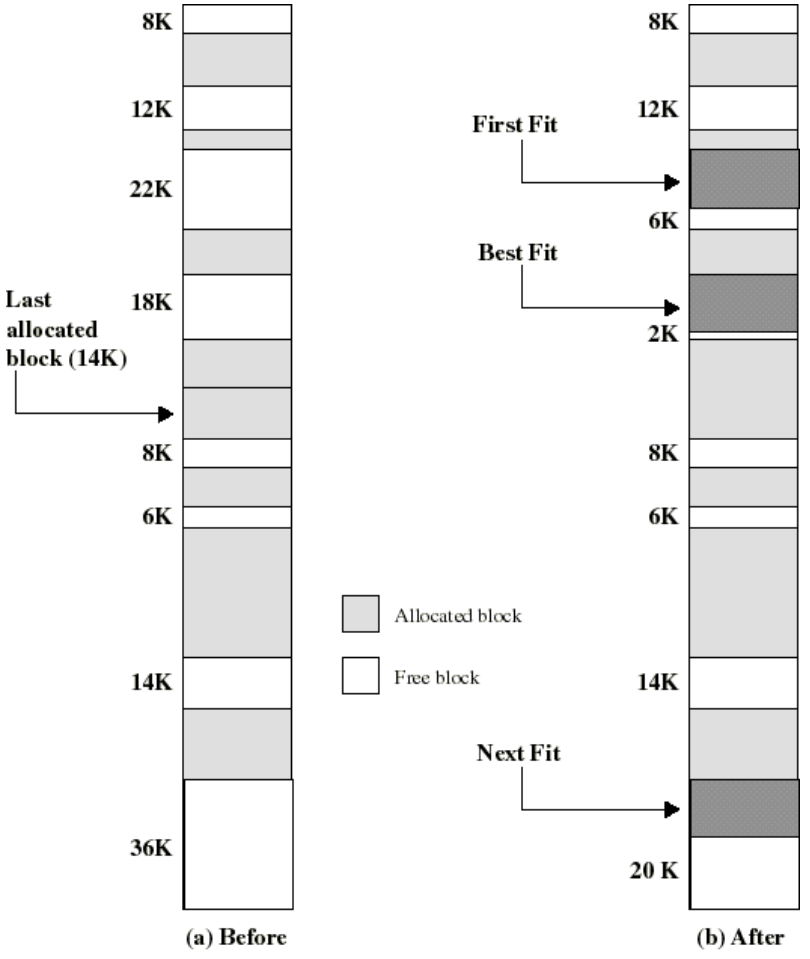
Dynamic partitioning

- Số lượng partition không cố định và partition có thể có kích thước khác nhau
- Mỗi process được cấp phát chính xác dung lượng bộ nhớ cần thiết
- Gây ra hiện tượng phân mảnh ngoại



Chiến lược placement

- Dùng để quyết định cấp phát khối bộ nhớ trống nào cho một process
- Mục tiêu: giảm chi phí compaction
- Các chiến lược placement
  - Best-fit: chọn khối nhớ trống nhỏ nhất
  - First-fit: chọn khối nhớ trống phù hợp đầu tiên kể từ đầu bộ nhớ
  - Next-fit: chọn khối nhớ trống phù hợp đầu tiên kể từ vị trí cấp phát cuối cùng
  - Worst-fit: chọn khối nhớ trống lớn nhất



Example Memory Configuration Before and After Allocation of 16 Kbyte Block

COMPUTER ENGINEERING

Tóm tắt lại nội dung buổi học

- Khái niệm cơ sở
- Các kiểu địa chỉ nhớ
- Chuyển đổi địa chỉ nhớ
- Overlay và swapping
- Mô hình quản lý bộ nhớ

COMPUTER ENGINEERING

Bài tập 1

Giả sử bộ nhớ chính được cấp phát các phân vùng có kích thước là 600K, 500K, 200K, 300K (theo thứ tự), sau khi thực thi xong, các tiến trình có kích thước 212K, 417K, 112K, 426K (theo thứ tự) sẽ được cấp phát bộ nhớ như thế nào, nếu sử dụng: Thuật toán First fit, Best fit, Next fit, Worst fit? Thuật toán nào cho phép sử dụng bộ nhớ hiệu quả nhất trong trường hợp trên

Bài tập 2