HỆ ĐIỀU HÀNH

ÔN TẬP CUỐI KỲ

PHAN ĐÌNH DUY

NỘI DUNG

1. Cấu trúc đề thi
2. Nội dung ôn thi
3. Ôn tập lý thuyết - bài tập
4. Giải đề thi mẫu
5. Hỏi – đáp – thảo luận

1. Cấu trúc đề thi

- Trắc nghiệm: 20 câu – 6.0 điểm (0.3 điểm/câu)
- Tự luận: 2-3 câu – 4.0 điểm
- Thời gian làm bài: 80 phút
- Thi theo lịch của phòng ĐT (10/01/2025 ca 2)

2. Nội dung ôn thi (tt)

- Chương 5: Đồng bộ tiến trình
- Chương 6: Deadlock
- Chương 7: Quản lý bộ nhớ
- Chương 8: Bộ nhớ ảo
- Chương 9: Hệ điều hành Linux và Windows (<=2 câu TN)

3. Ôn tập lý thuyết - bài tập chương 5

- Race condition
- Vấn đề vùng tranh chấp
- Lời giải cho vấn đề vùng tranh chấp
- Các giải pháp phần mềm
- Giải pháp phần cứng
- Mutex locks

5.2 Vấn đề vùng tranh chấp

- Xem xét một hệ thống có n tiến trình $\{P_0, P_1, \ldots, P_{n-1}\}$
- Mỗi tiến trình có một **vùng tranh chấp** là một **đoạn code**:
  - Thực hiện việc thay đổi giá trị của dữ liệu được chia sẻ (có thể là các biến, bảng dữ liệu, file,...)
  - Khi một tiến trình đang thực hiện vùng tranh chấp của mình thì các tiến trình khác **KHÔNG** được thực hiện vùng tranh chấp của chúng.
- Vấn đề vùng tranh chấp chính là thiết kế cách thức xử lý các vấn đề trên.

5.3.1. Yêu cầu dành cho lời giải

- Lời giải cho bài toán vùng tranh chấp phải đảm bảo 03 yêu cầu sau:
  - (1) **Mutual exclusion** (loại trừ tương hỗ): Khi một tiến trình P đang thực thi trong vùng tranh chấp (CS) của nó thì không có tiến trình Q nào khác đang thực thi trong CS của Q.
  - (2) **Progress** (tiến triển): Một tiến trình tạm dừng bên ngoài vùng tranh chấp không được ngăn cản các tiến trình khác vào vùng tranh chấp.
  - (3) **Bounded waiting** (chờ đợi giới hạn): Mỗi tiến trình chỉ phải chờ để được vào vùng tranh chấp trong một khoảng thời gian có hạn định nào đó. Không xảy ra tình trạng đói tài nguyên (starvation).

5.3.2. Phân loại giải pháp

Phân loại theo sự hỗ trợ của phần cứng

Giải pháp phần mềm

- Không cần sự hỗ trợ từ phần cứng, có thể được thực hiện thông qua các kỹ thuật lập trình.
- Ví dụ: giải thuật Peterson, giải thuật Bakery, giải thuật Dekker.

Giải pháp dựa trên phần cứng

- Cần sự hỗ trợ của một vài phần cứng đặc biệt, ví dụ cung cấp cơ chế đơn nguyên cho một vài chỉ thị/lệnh nhất định.
- Ví dụ: Test & Set, Compare & Swap.

5.3.2. Phân loại giải pháp

Phân loại theo sự hỗ trợ của hệ điều hành

Busy waiting

- Không cần sự hỗ trợ của hệ điều hành.
- Sử dụng kỹ thuật lập trình để tiến trình/tiểu trình phải chờ đợi (trong khi liên tục kiểm tra điều kiện) để được vào vùng tranh chấp.

Sleep & Wake up

- Cần hệ điều hành cung cấp cơ chế (thông qua system call) để:
  - *Tạm dừng (block) tiến trình*: đưa tiến trình gọi lệnh này vào trạng thái ngủ (**sleep**) nếu không được vào vùng tranh chấp.
  - *Đánh thức tiến trình*: khi một tiến trình ra khỏi vùng tranh chấp, tiến trình này có thể "đánh thức" (**wake up**) một tiến trình khác đang ngủ để tiến trình đó vào vùng tranh chấp.

3. Ôn tập lý thuyết - bài tập chương 5 (tt)

- Semaphore
- Monitor
- Liveness
- Bài toán đồng bộ bounded-buffer
- Bài toán đồng bộ readers-writers
- Bài toán đồng bộ dining-philosophers

5.7.1. Định nghĩa semaphore

Định nghĩa thao tác **wait()**	Định nghĩa thao tác **signal()**

`wait(S) {`
`  while (S <= 0)`
`  ; // busy wait`
`  S--;`
`}`
	`signal(S) {`
	`  S++;`
	`}`

• Nếu semaphore S **không dương** thì tiến trình/tiểu trình phải chờ. • Khi tiến trình được thực hiện CS thì **trừ semaphore đi 1.** • Được sử dụng khi muốn **sử dụng tài nguyên.**	• Tăng giá trị của semaphore lên 1. • Được sử dụng khi **trả lại tài nguyên.**

5.7.3. Hiện thực semaphore

Hiện thực semaphore không busy waiting

Định nghĩa thao tác **wait()**	Định nghĩa thao tác **signal()**
`wait(S) { while (S <= 0) ; // busy wait S--; }`	`signal(S) { S++; }`
`wait(semaphore *S) { S->value--; if (S->value < 0) { add this process to S->list; block(); } }`	`signal(semaphore *S) { S->value++; if (S->value <= 0) { remove a process P from S->list; wakeup(P); } }`

3. Ôn tập lý thuyết - bài tập chương 6

- Vấn đề deadlock
  - Vấn đề deadlock
  - Định nghĩa
  - Điều kiện cần để xảy ra deadlock
- Mô hình hệ thống
  - Đồ thị cấp phát tài nguyen (RAG)
  - RAG và deadlock
- Phương pháp giải quyết deadlock

3. Ôn tập lý thuyết - bài tập chương 6 (tt)

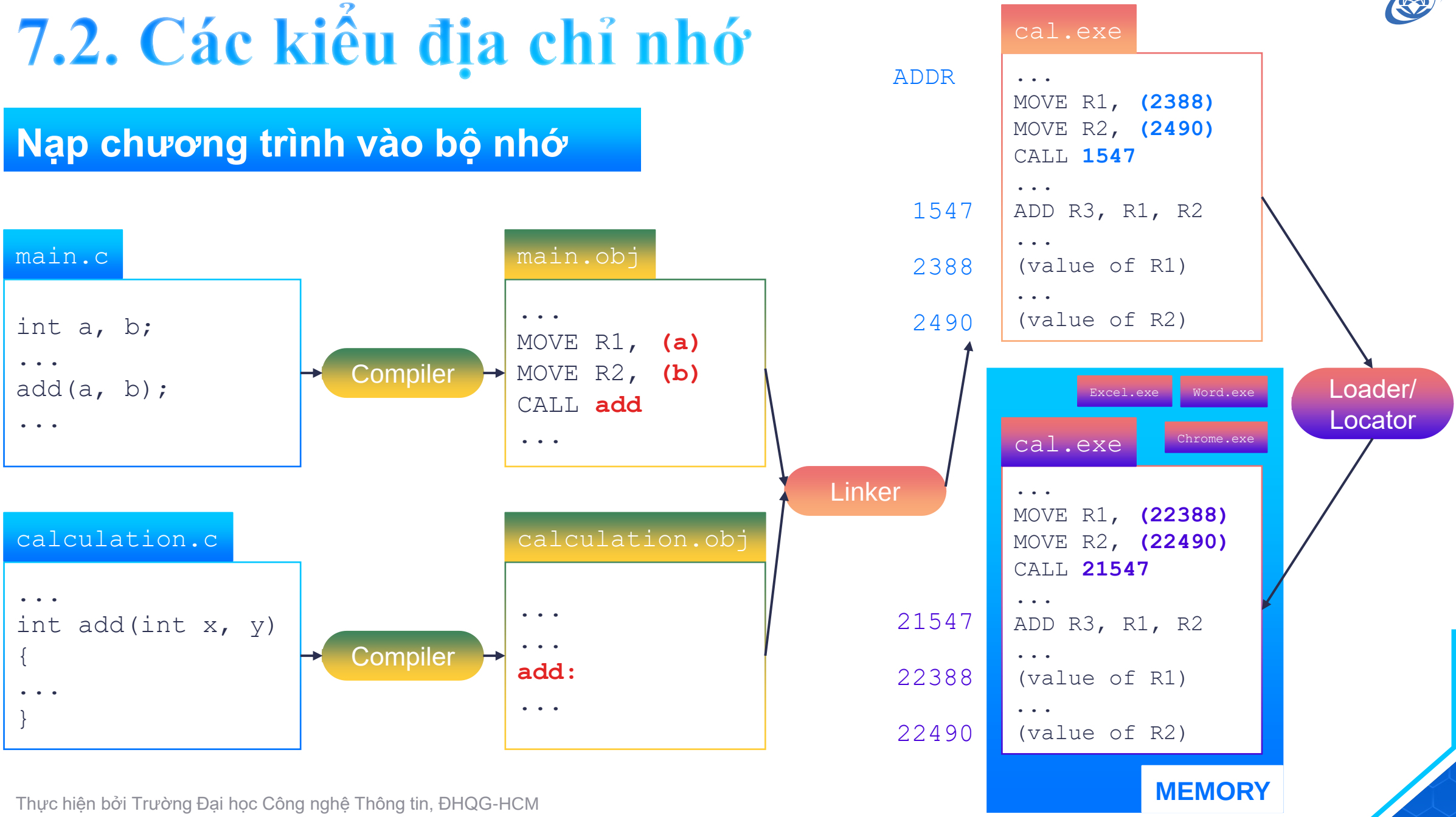
- Vấn đề deadlock
- Mô hình hệ thống
- Phương pháp giải quyết deadlock
  - Ngăn deadlock: không cho phép (ít nhất) một trong 4 điều kiện cần cho deadlock.
  - Tránh deadlock: các tiến trình cần cung cấp thông tin về tài nguyên nó cần để hệ thống cấp phát tài nguyên một cách thích hợp.
  - Cho phép hệ thống vào trạng thái deadlock, nhưng sau đó phát hiện deadlock và phục hồi hệ thống.
  - Bỏ qua mọi vấn đề, xem như deadlock không bao giờ xảy ra trong hệ thống.

3. Ôn tập lý thuyết - bài tập chương 7

- Khái niệm cơ sở
- Các kiểu địa chỉ nhớ
- Chuyển đổi địa chỉ nhớ
- Mô hình quản lý bộ nhớ
- Cơ chế phân trang
- Cơ chế swapping

7.2. Các kiểu địa chỉ nhớ

Nạp chương trình vào bộ nhớ

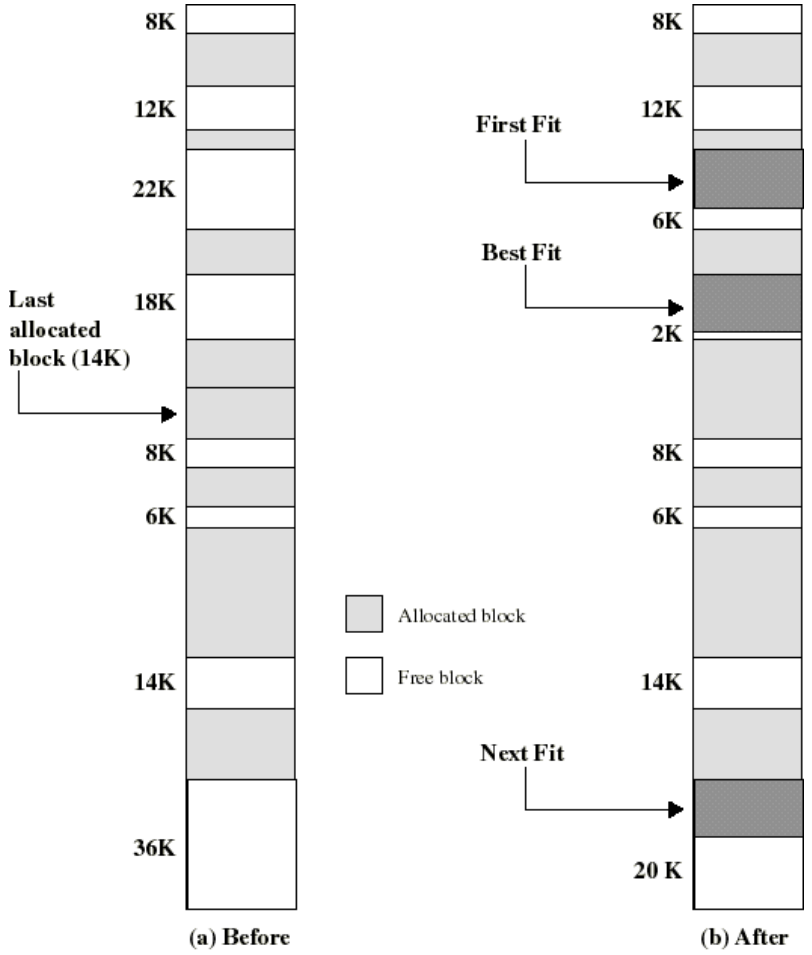


7.4. Mô hình quản lý bộ nhớ

- Trong chương này, mô hình quản lý bộ nhớ là một mô hình đơn giản, không có bộ nhớ ảo.
- Một tiến trình phải được nạp hoàn toàn vào bộ nhớ thì mới được thực thi.
- Các cơ chế quản lý bộ nhớ:
  - Phân chia cố định (fixed partitioning)
  - Phân chia động (dynamic partitioning)
  - Phân trang đơn giản (simple paging)
  - Phân đoạn đơn giản (simple segmentation)

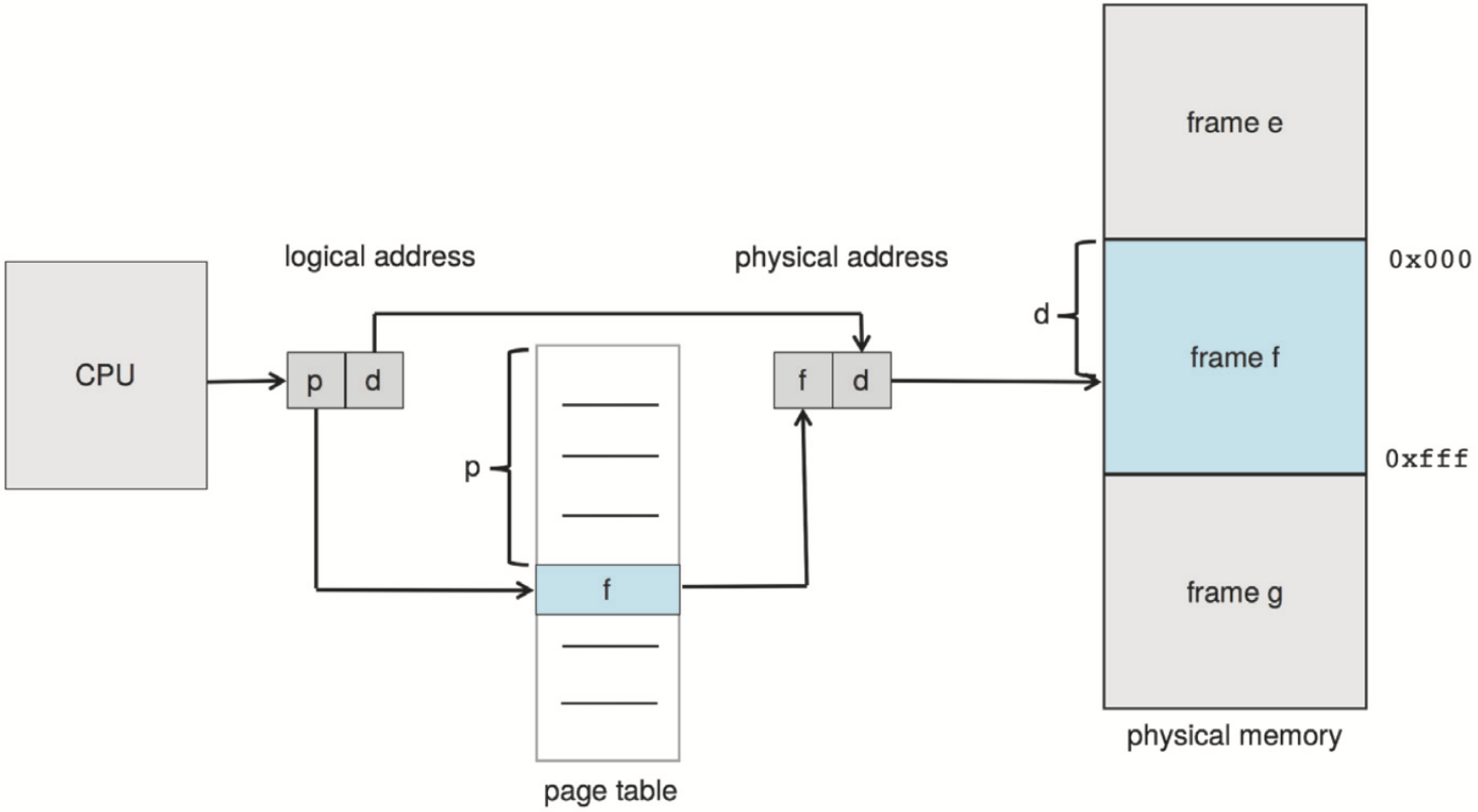
7.4.2. Dynamic partitioning - Chiến lược placement

- Dùng để quyết định cấp phát khối bộ nhớ trống nào cho một tiến trình.
- 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.



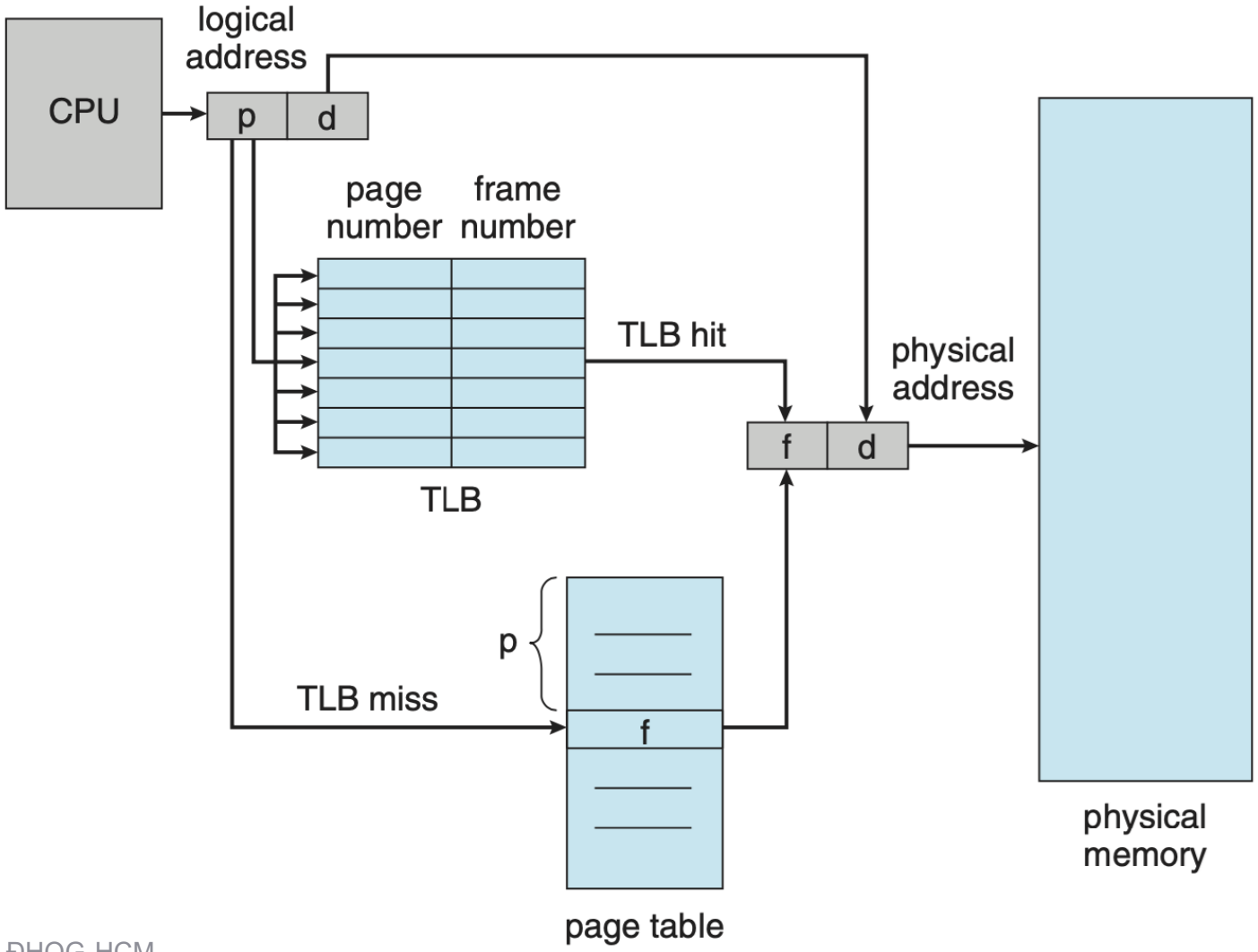
Example Memory Configuration Before and After Allocation of 16 Kbyte Block

7.5.1. Chuyển đổi địa chỉ trong paging



7.5.2. Cài đặt bảng trang

Dùng TLB



3. Ôn tập lý thuyết - bài tập chương 8

- Tổng quan về bộ nhớ ảo
- Cài đặt bộ nhớ ảo: Demand Paging
- Các giải thuật thay trang (Page Replacement Algorithms)
- Vấn đề cấp phát Frames
- Vấn đề Thrashing

8.3.1 Các giải thuật thay trang

- Giải thuật thay trang FIFO
- Giải thuật thay trang OPT
- Giải thuật thay trang LRU
- Các dữ liệu cần biết ban đầu:
  - Số khung trang
  - Tình trạng ban đầu
  - Chuỗi tham chiếu

Bài tập chương 8

Xét chuỗi truy xuất bộ nhớ sau:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1

Có bao nhiêu lỗi trang xảy ra khi sử dụng các thuật toán thay thế sau đây, giả sử hệ thống có 4 khung trang.

a. LRU

b. FIFO

c. Chiến lược tối ưu (OPT)

4. Giải đề thi mẫu

- Đề thi cuối HK2 năm 2023-2024

5. Hỏi – đáp – thảo luận

- …

THẢO LUẬN