SISTEMAS OPERACIONAIS

AULA 15 – REVISÃO GERENCIAMENTO DE MEMÓRIA E EXERCÍCIOS

Prof.ª Sandra Cossul, Ma.

INTRODUÇÃO

- **Memória principal** (RAM) → programas em execução
- **Memória secundária** (HD, SSD) → mecanismos de armazenamento permanente com maior capacidade

- **Para um programa ser executado, deve ser carregado na memória principal!**

- **A eficiência da multiprogramação exige que vários programas estejam na memória ao mesmo tempo, vindo a necessidade de organização da memória.**

ALOCAÇÃO DE MEMÓRIA

1. **Alocação Particionada**
   A. Estática (tamanho fixo e variável)
   B. Dinâmica

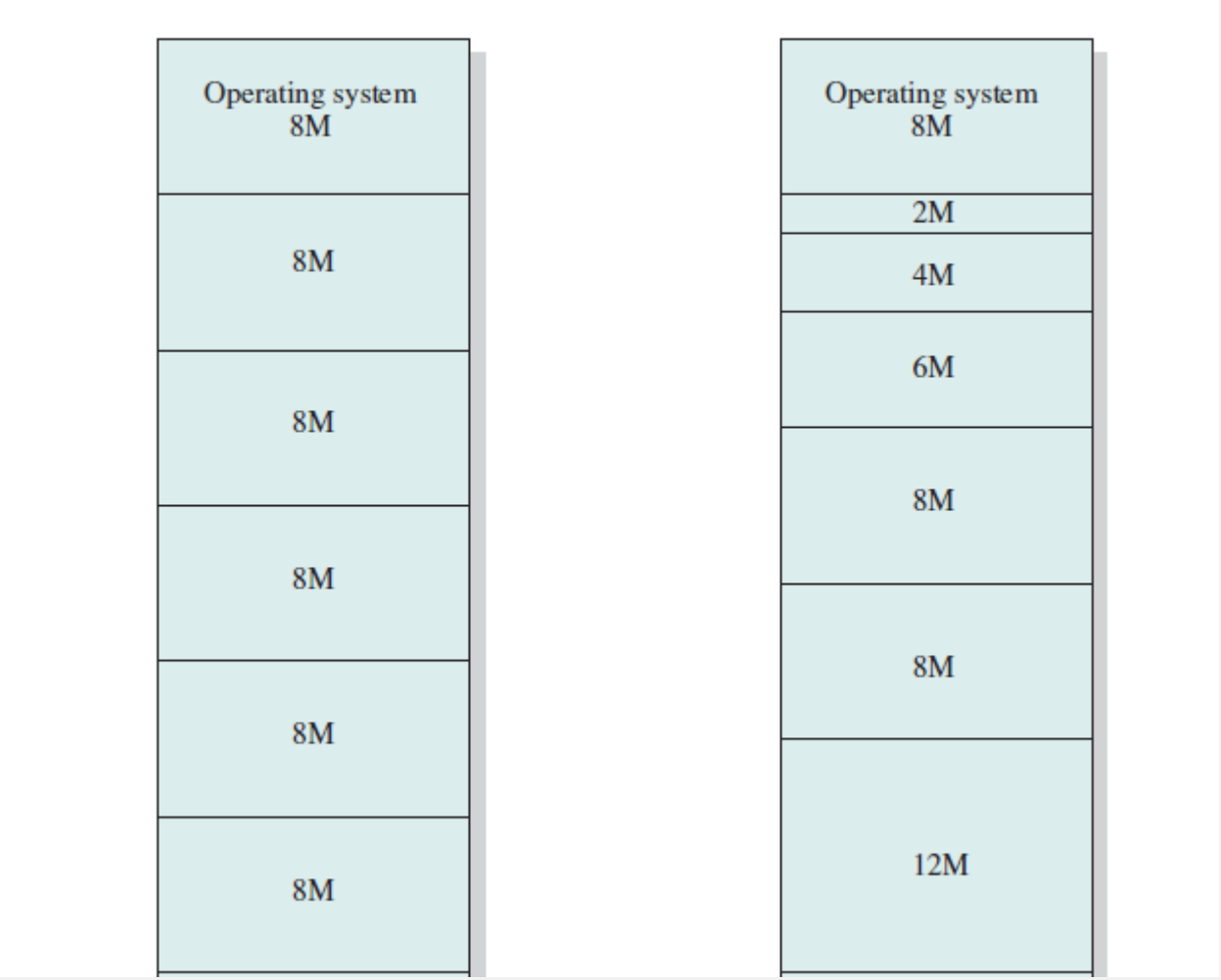
2. **Paginação**
3. **Segmentação**
4. **Memória virtual**

1. A) ALOCAÇÃO PARTICIONADA ESTÁTICA

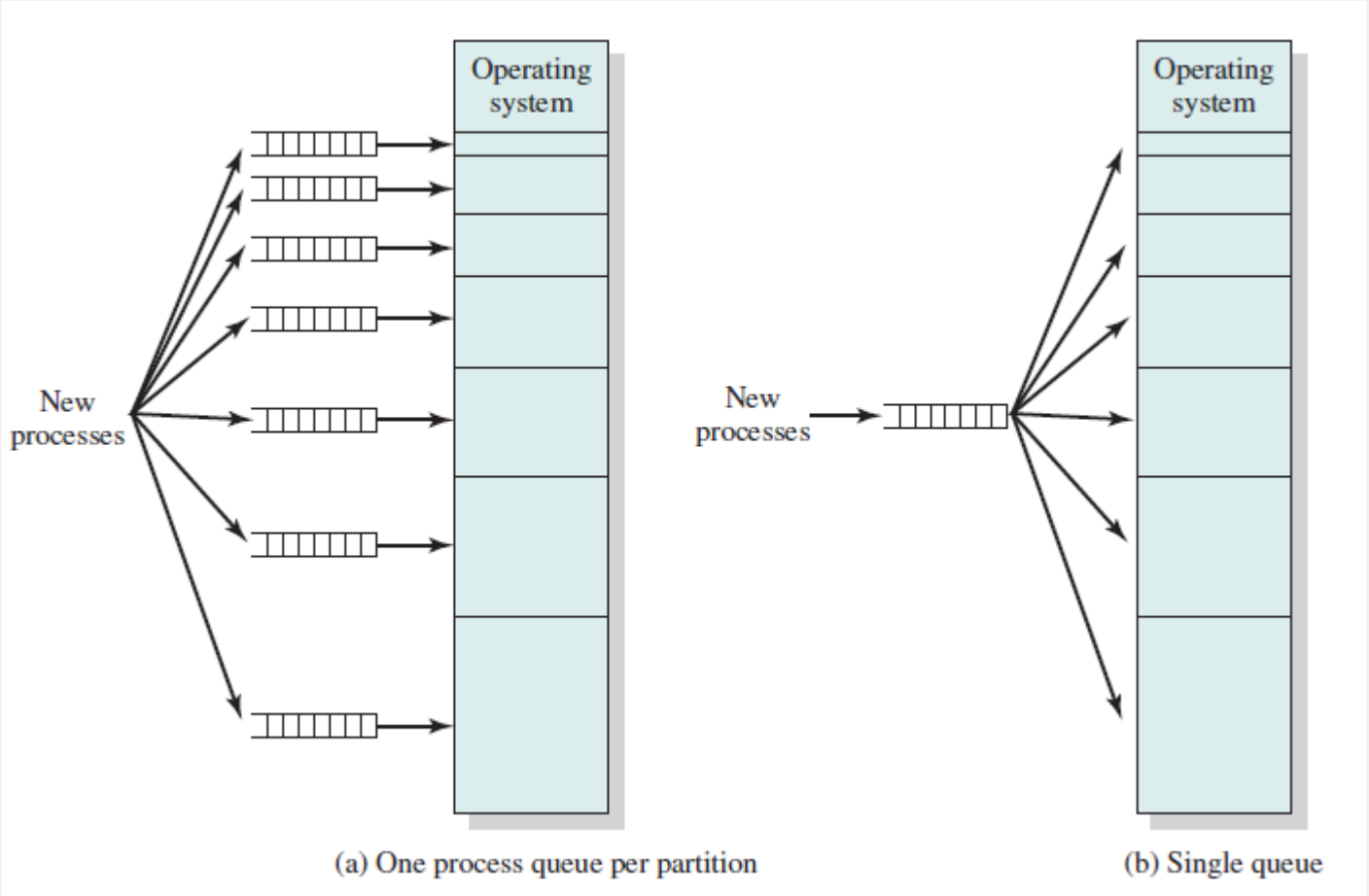
- **Divisão da memória em tamanhos fixos** (partições), definidos na inicialização do SO

- Estas partições podem ser inicializadas com **tamanho fixo** ou **tamanho variável**
  - fila de escalonamento para cada partição (escolha da menor partição disponível)
  - OU única fila para todas as partições

1. A) ALOCAÇÃO PARTICIONADA ESTÁTICA



1.A) ALOCAÇÃO PARTICIONADA ESTÁTICA



1. A) ALOCAÇÃO PARTICIONADA ESTÁTICA

- Programas não preenchem totalmente as partições onde são carregados.
- Problemas de **fragmentação interna**.

- Número de partições **limitam** o número de **processos ativos** na memória

- Técnica não mais utilizada nos SOs modernos!

1. B) ALOCAÇÃO PARTICIONADA DINÂMICA

- Partições tem **tamanho e número variado**
- Partições sem tamanho fixo, onde cada programa utiliza o **espaço que necessita**

- Aumento do grau de compartilhamento diminuindo o problema da fragmentação interna

- No entanto, ocorre a **fragmentação externa,** em que conforme os programas vão terminando vão deixando espaços cada vez menores.

I. B) ALOCAÇÃO PARTICIONADA DINÂMICA

- **Soluções fragmentação externa:**
  - **Compactação -** Reunir os espaços adjacentes, produzindo um único espaço de tamanho maior (processo custoso!)

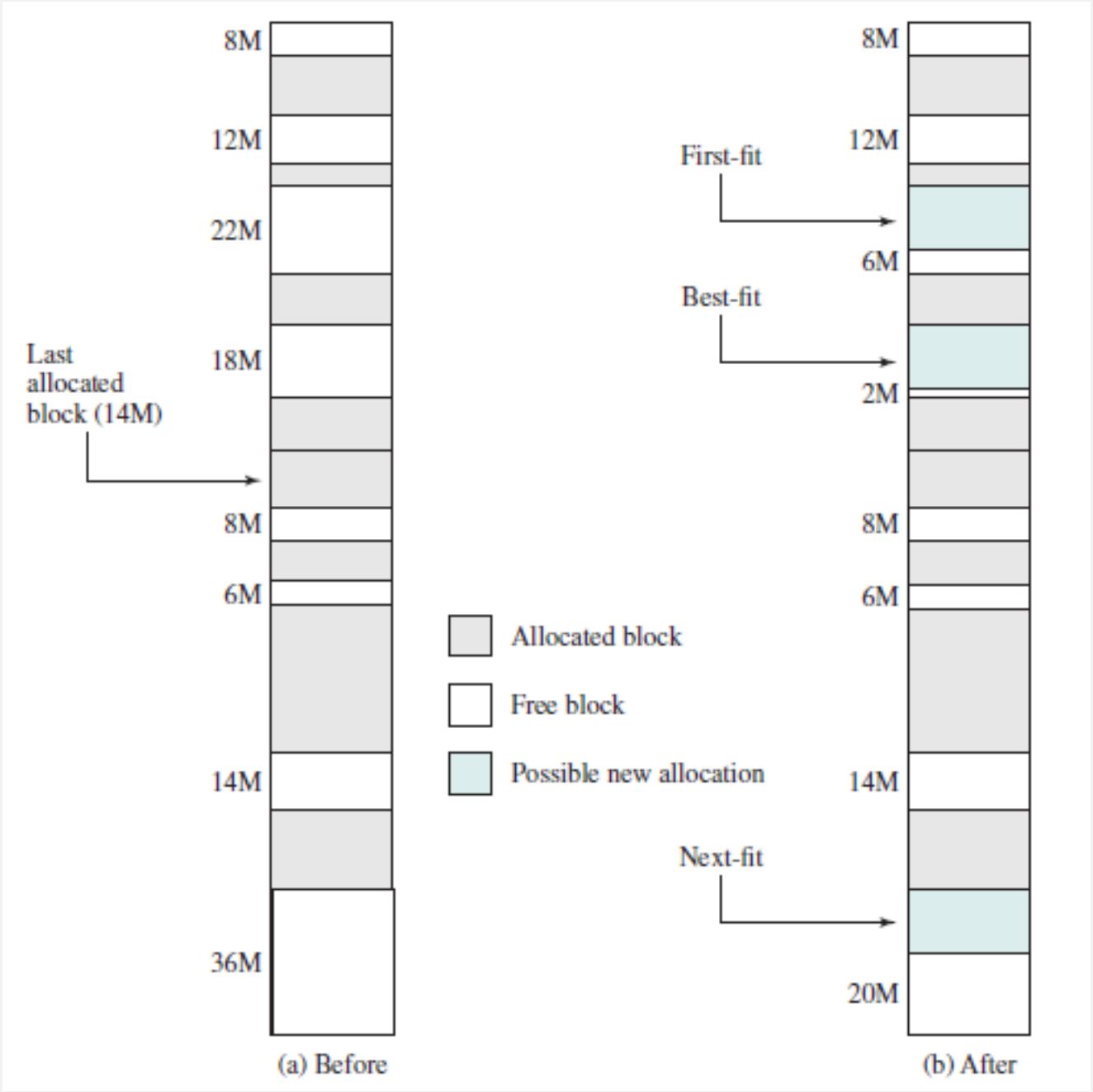
- **Como evitar ou diminuir o problema de fragmentação antes que ele ocorra?**
  - Criando estratégias para determinar em qual partição livre um programa será carregado para execução

I. B) ALOCAÇÃO PARTICIONADA DINÂMICA

- **Estratégias para Escolha da Partição:**
  - ***Best- fit*:** escolhe o bloco de tamanho mais próximo ao requisitado
  - ***First fit*:** percorre a memória desde o início e escolhe o primeiro bloco disponível que tem tamanho suficiente
  - ***Next fit*:** percorre a memória a partir do último local de atribuição e escolhe o próximo bloco disponível que tem tamanho suficiente

ESTRATÉGIAS PARA ESCOLHA DA PARTIÇÃO

→ Alocar um processo de 16MB



CONCEITOS IMPORTANTES

- **Memória lógica (virtual) -** memória que o processo enxerga ou seja, aquela que é capaz de endereçar e acessar usando as suas instruções
- **Espaço de endereçamento lógico (virtual) –** formado pelos endereços lógicos que um processo pode gerar

- **Memória física –** implementado pela memória física (hardware)
- **Espaço de endereçamento físico (real) –** endereços com localização real na memória principal.

- **Mapeamento** → tradução do endereço virtual para físico

CONCEITOS IMPORTANTES

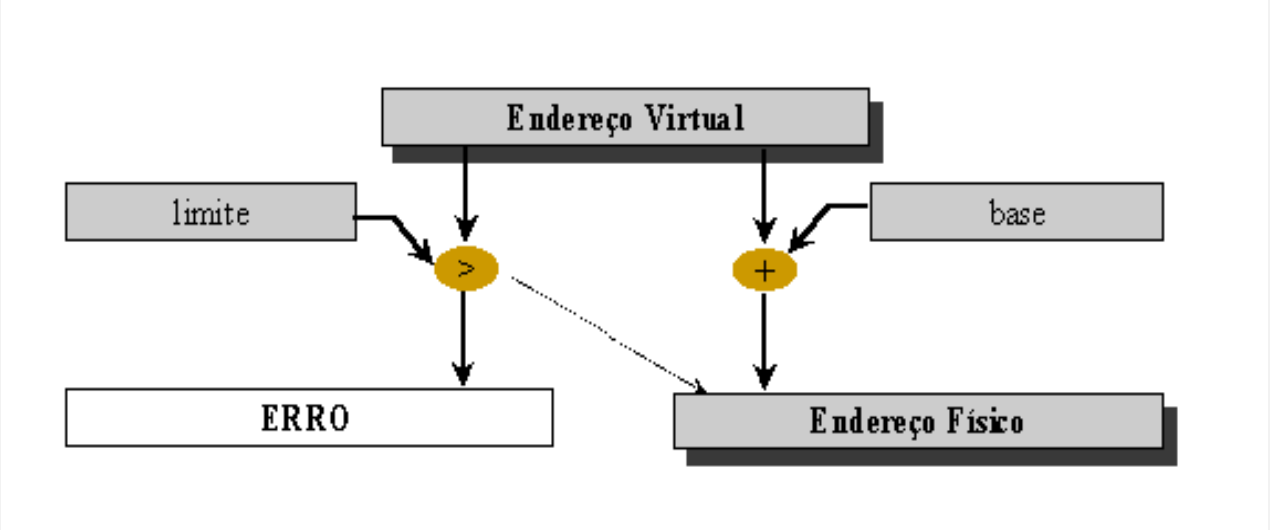
- **Mapeamento →** tradução do endereço virtual para físico
  - **Todo programa precisa estar no espaço de endereçamento real para poder ser referenciado ou executado;**
  - É realizado via **hardware** junto com o **Sistema Operacional**, de forma a não comprometer seu desempenho e torná-lo transparente aos usuários e suas aplicações;
  - Memória associativa ou **Translation Lookside Buffer** – Hardware especial para mapear endereços virtuais para endereços físicos sem a necessidade de acesso à tabelas de páginas;

CONCEITOS IMPORTANTES

- **Unidade de gerência de memória (MMU)**
  - Componente do hardware responsável por prover os mecanismos básicos que serão usados pelo SO para gerenciar a memória.
  - A MMU realiza o mapeamento dos endereços lógicos gerados pelos processos para os endereços físicos correspondentes

CONCEITOS IMPORTANTES

- **Realocação de processos**
  - Proteção da memória através de dois registradores, início e fim da partição.
  - Quando um processo é escalonado o **registrador-base** é carregado com o endereço de **início da partição** e o **registrador-limite** com o **tamanho da partição**



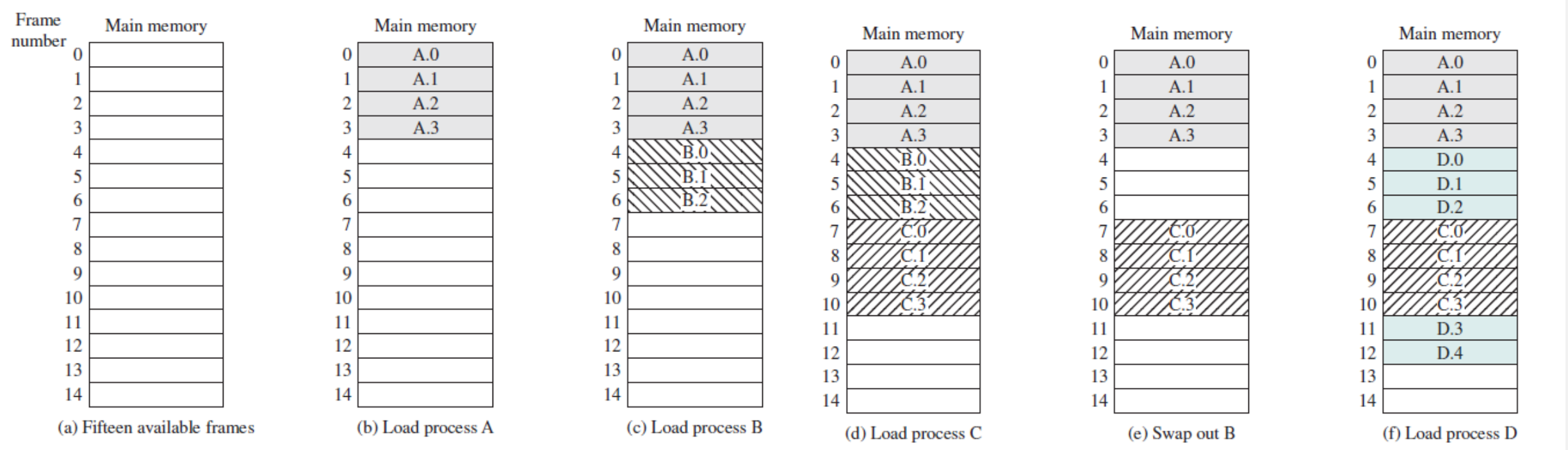
2) PAGINAÇÃO

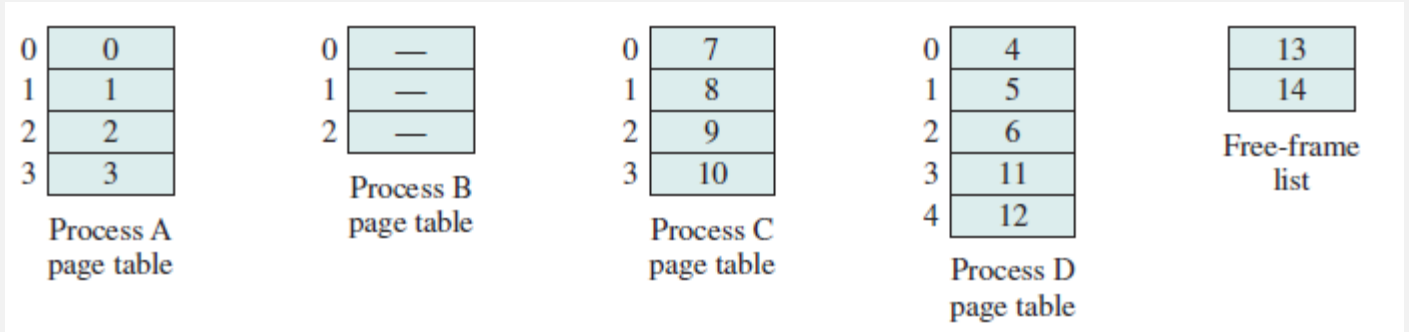
- Técnica de gerência de memória que permite que o **espaço de endereçamento real** (físico) de um processo seja **não contíguo**

- A memória principal é dividida em blocos de tamanho fixo → **frames**
- A memória lógica é dividida em blocos do mesmo tamanho → **páginas**
  - Memória lógica se refere aos processos, os quais são divididos em páginas!

- **Os "pedaços" de processos (páginas) são designados a "pedaços" livres de memória (frames)!**

2) PAGINAÇÃO – IMPLEMENTAÇÃO

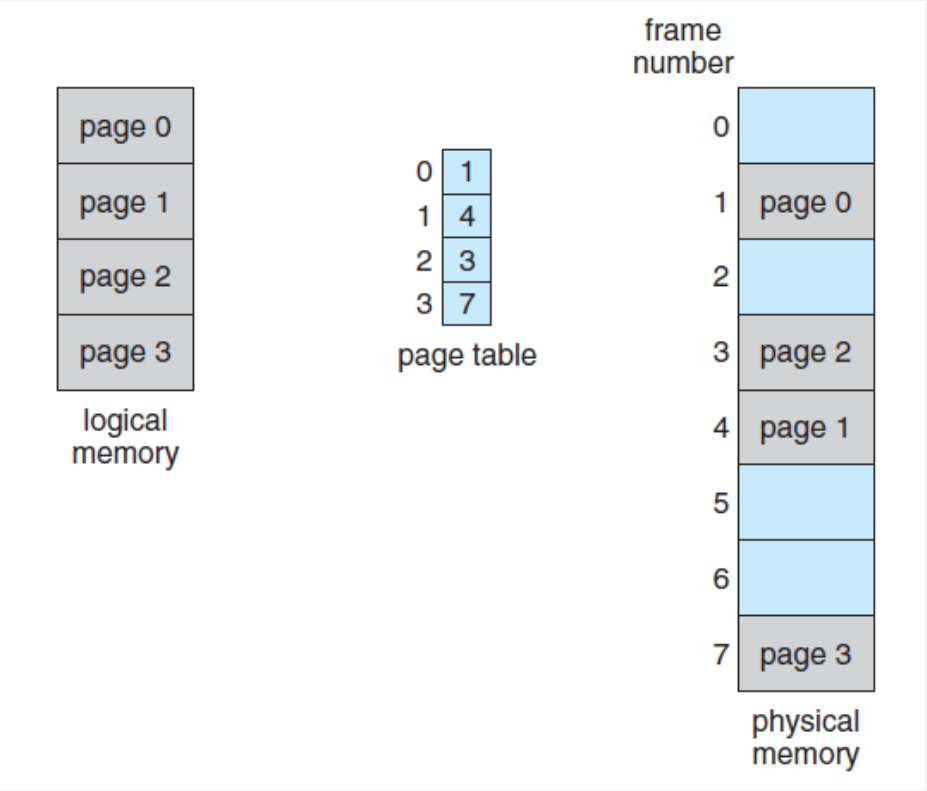




2) PAGINAÇÃO

- Com paginação, não acontece fragmentação: qualquer frame pode ser alocado a um processo (ou uma página de processo) conforme for preciso.

- Uma **tabela de páginas** é criada pelo SO para **gerenciar os endereços.**
  - Cada entrada uma página do processo → número do frame correspondente



3) SEGMENTAÇÃO

- Técnica de gerência de memória, onde os programas são divididos logicamente e em sub-rotinas e estruturas de dados e colocados em **blocos de informações** na memória
- **Segmentos** – blocos de tamanhos diferentes com seu próprio espaço de endereçamento (tabela de segmentos).

- **Segmentação X Paginação** – Paginação com partes de tamanho fixo e segmentos com blocos de tamanhos variados
- Permite uma relação entre a lógica do programa e sua divisão na memória

SEGMENTAÇÃO E PAGINAÇÃO

- Em geral, a **segmentação** é usada juntamente com **paginação**
  - **A segmentação possibilita um mecanismo "alto-nível" para o usuário**
  - **A paginação provê um mecanismo automático de alocação**

4) MEMÓRIA VIRTUAL

- Combina **memória principal** e **secundária**;
- Impressão da memória ser muito maior do que é;
- Desvinculação do endereçamento feito pelo programa dos endereços físicos da memória principal;
- Procura minimizar o problema de fragmentação da memória.

- Apenas parte do programa pode estar residente na memória em um determinado instante;
- O SO utiliza a **memória secundária** como uma **extensão da memória principal.**

4) MEMÓRIA VIRTUAL

- Combina **memória principal** e **secundária**;
- Desvinculação do endereçamento feito pelo programa dos endereços físicos da memória principal;
  - Memória principal como **extensão da memória secundária**
  - Impressão da memória ser muito maior do que é;
- Procura minimizar o problema de fragmentação da memória.

4) MEMÓRIA VIRTUAL

- Utilizada em conjunto com **paginação** e **segmentação**

- Nem todas as páginas (ou segmentos) de um processo precisam estar na memória principal para ele executar.
- Páginas (ou segmentos) podem ser trazidas conforme necessário.

- **Paginação por demanda** é quando as páginas dos processos são transferidas da memória secundária para a principal apenas quando são referenciadas, seguindo o princípio da localidade.

4) MEMÓRIA VIRTUAL

- **Problemas:**
  - Paginação exige operações de E/S (que deve ser evitado) quando um processo faz referência a uma página que não se encontra na memória;
  - O SO deve se preocupar em ter um certo número de páginas na memória que reduza ao máximo a taxa de paginação dos processos e não prejudique os demais processos que desejam acesso a memória.

- **Conjunto residente (working set) –** conjunto de páginas presentes na memória principal num certo intervalo de tempo!

4) MEMÓRIA VIRTUAL

- **Princípio da Localidade**
  - **Espacial:** tendência que existe em um programa de fazer referências a posições de memória de forma quase uniforme, ou seja, instruções próximas.
  - **Temporal:** tendência que existe em um programa de fazer referências a posições de memória utilizadas recentemente

4) MEMÓRIA VIRTUAL

- **Thrashing –** excessiva transferência de páginas/segmentos entre a memória principal e a memória secundária.
- **Na Paginação ou segmentação:**
  - Conjunto residente pode ser pequeno demais para acomodar as páginas referenciadas (aumentar o cjto residente)
  - Não obediência do princípio da localidade – programa faz referências a comandos/dados localizados em páginas fora do cjto residente (reescrever a aplicação)
  - Transferência excessiva devido a modularização extrema do programa
  - Em nível de sistema, ocorre quando existem mais processos competindo por memória que espaço disponível

4) MEMÓRIA VIRTUAL
ESTRATÉGIAS DE REALOCAÇÃO DE PÁGINAS

- **Realocação de páginas**
  - Problema em decidir quais páginas remover da memória principal.
  - O SO deve considerar se uma página foi ou não modificada antes de liberá-la para outro processo, caso contrário, possíveis dados armazenados na página serão perdidos.
  - Sempre que uma página é alterada, um bit de modificação é alterado de 0 para 1, informando que a página foi alterada.
  - Melhor estratégia de realocação é escolher uma página que não será referenciada num futuro próximo. Tarefa difícil para o Sistema Operacional.

4) MEMÓRIA VIRTUAL
ESTRATÉGIAS DE REALOCAÇÃO DE PÁGINAS

- **Ótimo**
  - seleciona para substituir a página que vai ter o acesso mais demorado
  - Melhor algoritmo, mas impossível de implementar (não tem como prever o futuro e saber quantas instruçoes faltam para a página ser acessada)

- **FIFO (first-in first-out)**
  - A página que primeiro foi utilizada será a primeira a ser escolhida.
  - Implementação bastante simples.
  - Necessário apenas uma fila.

4) MEMÓRIA VIRTUAL
ESTRATÉGIAS DE REALOCAÇÃO DE PÁGINAS

- **LRU (least recently used)**
  - Seleciona a página utilizada menos recentemente, ou seja, a que está há mais tempo sem ser referenciada.
  - Estratégia boa, mas pouco implementada;
  - Grande overhead causado pela atualização, em cada página referenciada, do momento do último acesso.
  - **Opção:** LRU utilizando matrizes!

PROTEÇÃO MEMÓRIA

- Necessária para impedir que um processo, ao acessar uma página/segmento do sistema, a modifique ou mesmo tenha acesso a ela.

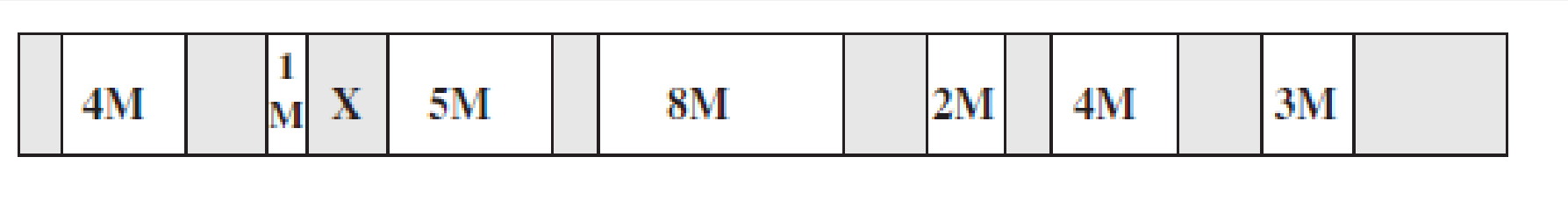
- No esquema de memória virtual, cada processo tem sua própria tabela de mapeamento e a tradução dos endereços é realizada pelo sistema, impedindo assim, que um processo tenha acesso a áreas de memória de outros processos, a não ser que tenham compartilhamento explícito.

- A proteção deve ser realizada em nível de cada página/segmento na memória, utilizando-se as entradas da tabela de mapeamento, com alguns bits especificando permissões a cada uma das páginas/segmentos.

EXERCÍCIOS

1) O diagrama abaixo mostra uma configuração de memória utilizando particionamento dinâmico, após alguns processos já foram alocados e liberados da memória.

- **Áreas em cinza** → áreas ocupadas por processos
- **Áreas em branco** → áreas livres
- O último processo alocado tem 2 Mb e está marcado com um X.
- Um novo processo de **3Mb** deve ser alocado em seguida. Indique onde será alocado utilizando: **best-fit, first-fit e next-fit.**



EXERCÍCIOS

2) Considere uma memória com 6 partições:

300 KB	600 KB	350 KB	200 KB	750 KB	125KB

a) Como os algoritmos **first-fit** e **best-fit** alocariam a memória para os processos a seguir (em ordem):

P1 115 KB, P2 500 KB, P3 358 KB, P4 200 KB e P5 375KB?

EXERCÍCIOS

3) Considerando as seguintes referências de páginas:

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

→ Quantas **page faults** vão acontecer considerando 1, 2, 3, 4, 5, 6 e 7 frames e os seguintes algoritmos de substituição de páginas?

a) **LRU**
b) **FIFO**
c) **Ótimo**

Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

EXERCÍCIOS

4) Considerando a seguinte sequência de requisições de páginas e que existem apenas **três frames** na memória RAM para alocar as páginas virtuais, simule a execução dos seguintes algoritmos de troca de páginas e **calcule a quantidade de falta de páginas** em cada um:

a) Ótimo

b) FIFO

c) LRU

Seqüência: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

OBS: Inicialmente, todas os frames estão vazios, ou seja, nenhuma página está carregada na memória.

BIBLIOGRAFIA

- Tanenbaum, A. S. **Sistemas Operacionais Modernos.** Pearson Prentice Hall. 3rd Ed., 2009.
- Silberschatz, A; Galvin, P. B.; Gagne G.; **Fundamentos de Sistemas Operacionais.** LTC. 9th Ed., 2015.
- Stallings, W.; **Operating Systems: Internals and Design Principles**. Prentice Hall. 5th Ed., 2005.
- Oliveira, Rômulo, S. et al. **Sistemas Operacionais** - V11 - UFRGS. Disponível em: Minha Biblioteca, Grupo A, 2010.