SISTEMAS OPERACIONAIS

AULA 13 – GERENCIAMENTO DE MEMÓRIA PAGINAÇÃO, SEGMENTAÇÃO – PARTE 2

Prof.ª Sandra Cossul, Ma.

RELEMBRANDO...

- A memória principal contém tanto o **sistema operacional** quanto os **vários processos de usuário.**
- A memória deve ser alocada da forma **mais eficiente possível.**
- Vários **processos** estão na **memória ao mesmo tempo**
  - Como alocar memória disponível aos processos que estão esperando para ser trazidos para a memória? **Gerenciamento de memória!**

RELEMBRANDO... PARTICIONAMENTO DA MEMÓRIA

- **Particionamento fixo**
  - **Partições fixas**
  - **Partições variáveis**

- **Particionamento dinâmico**
  - **Partições que se adaptam ao tamanho do processo**
  - Algoritmos de alocação dos processos na memória:
    - **Best-fit, First-fit e Next-fit**

RELEMBRANDO... FRAGMENTAÇÃO

- **Fragmentação interna**
  - Memória não utilizada dentro de uma partição
  - Memória alocada a um processo maior do que o necessário devido ao tamanho da partição

- **Fragmentação externa**
  - Memória não utilizada externa às partições
  - Memória vai ficando com vários "buracos" de espaço livre

RELEMBRANDO... FRAGMENTAÇÃO

- **Solução para fragmentação: Compactação**
  - "juntar" o conteúdo de memória utilizado em um único grande bloco
  - processo custoso que nem sempre é possível

- **Outra Solução para fragmentação: Paginação**
  - permitir que a alocação da memória seja **não-contígua**
  - alocar memória onde estiver disponível

PAGINAÇÃO

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

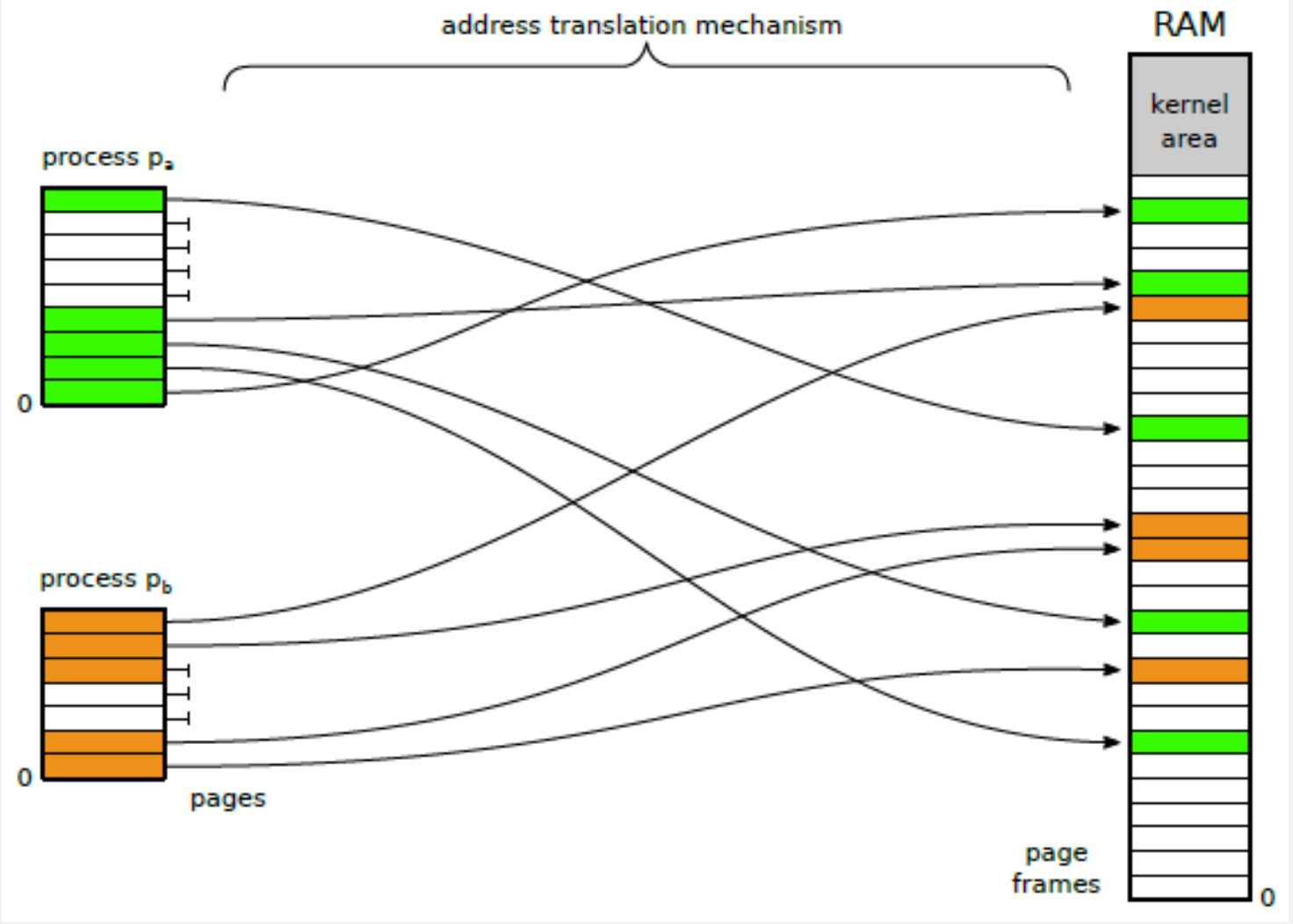
- Evita fragmentação e a necessidade associada de compactação.

PAGINAÇÃO – IMPLEMENTAÇÃO

- 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)!**

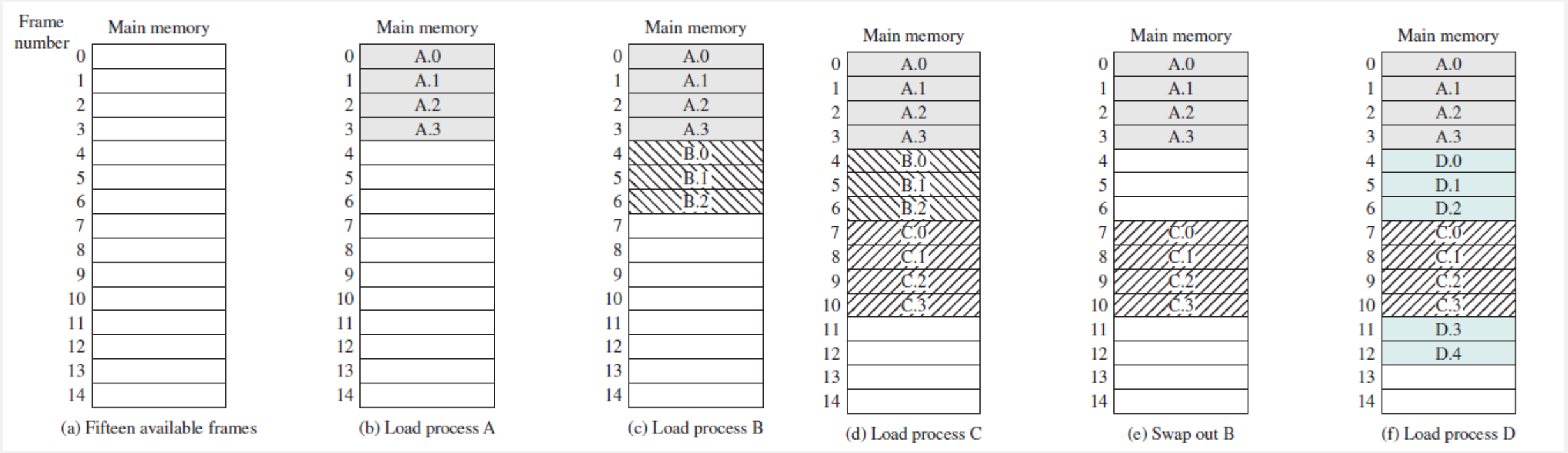
PAGINAÇÃO – IMPLEMENTAÇÃO

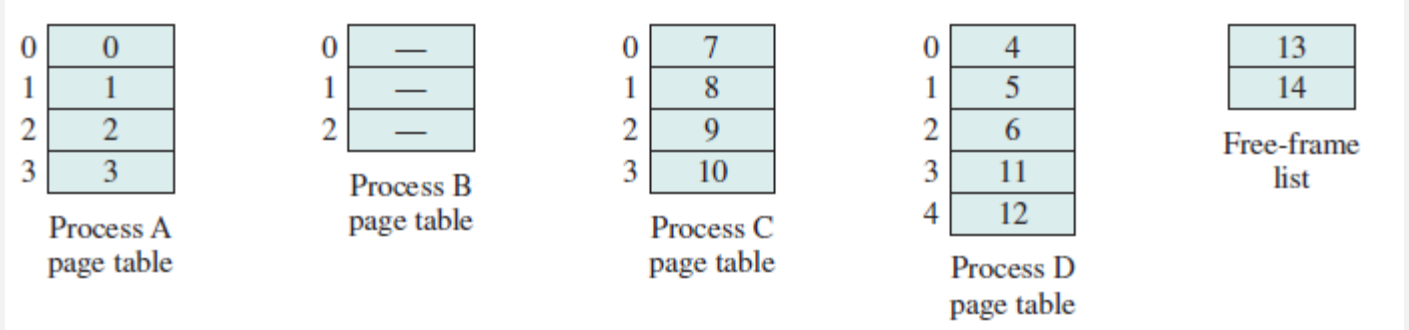


PAGINAÇÃO – IMPLEMENTAÇÃO

- **Exemplo**
  - Em um dado instante, os frames da memória podem estar **livres** ou **ocupados**
  - Uma lista de frames livres é mantida pelo SO
  - **Processos A, B e C** (mantidos na memória secundária) tem **4, 3 e 4 páginas**
  - Processos A, B e C são carregados na memória
  - Processo B é suspenso e trocado (swapping) e SO precisa trazer um novo processo D (5 páginas)
  - **O que acontece já que não existem 5 frames livres de memória sequenciais?**
    - Utiliza-se **paginação** e o processo D é dividido entre os frames livres!

PAGINAÇÃO – IMPLEMENTAÇÃO





PAGINAÇÃO – IMPLEMENTAÇÃO

- **Como o SO controla a divisão dos processos na memória principal?**
  - Mantendo uma **tabela de páginas** para cada processo!
  - Essa tabela guarda a **localização** de cada frame para cada página do processo.

- Dentro do programa, cada **endereço lógico** consiste em um **número de página** e um **offset** dentro da página.

Número da página	Offset

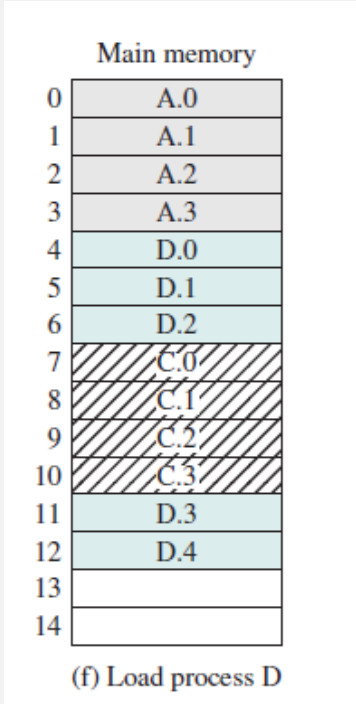
- O CPU utiliza a tabela de páginas para produzir um endereço físico

Número do frame	Offset

PAGINAÇÃO – IMPLEMENTAÇÃO

- A **tabela de páginas** contém uma **entrada para cada página do processo**
- Indexada pelo número da página (iniciando em 0)
- Cada entrada da tabela de páginas contém o **número do frame** na memória principal, se houver, que contém a página correspondente.

- Além disso, o SO mantém uma **lista de frames livres** (desocupados)



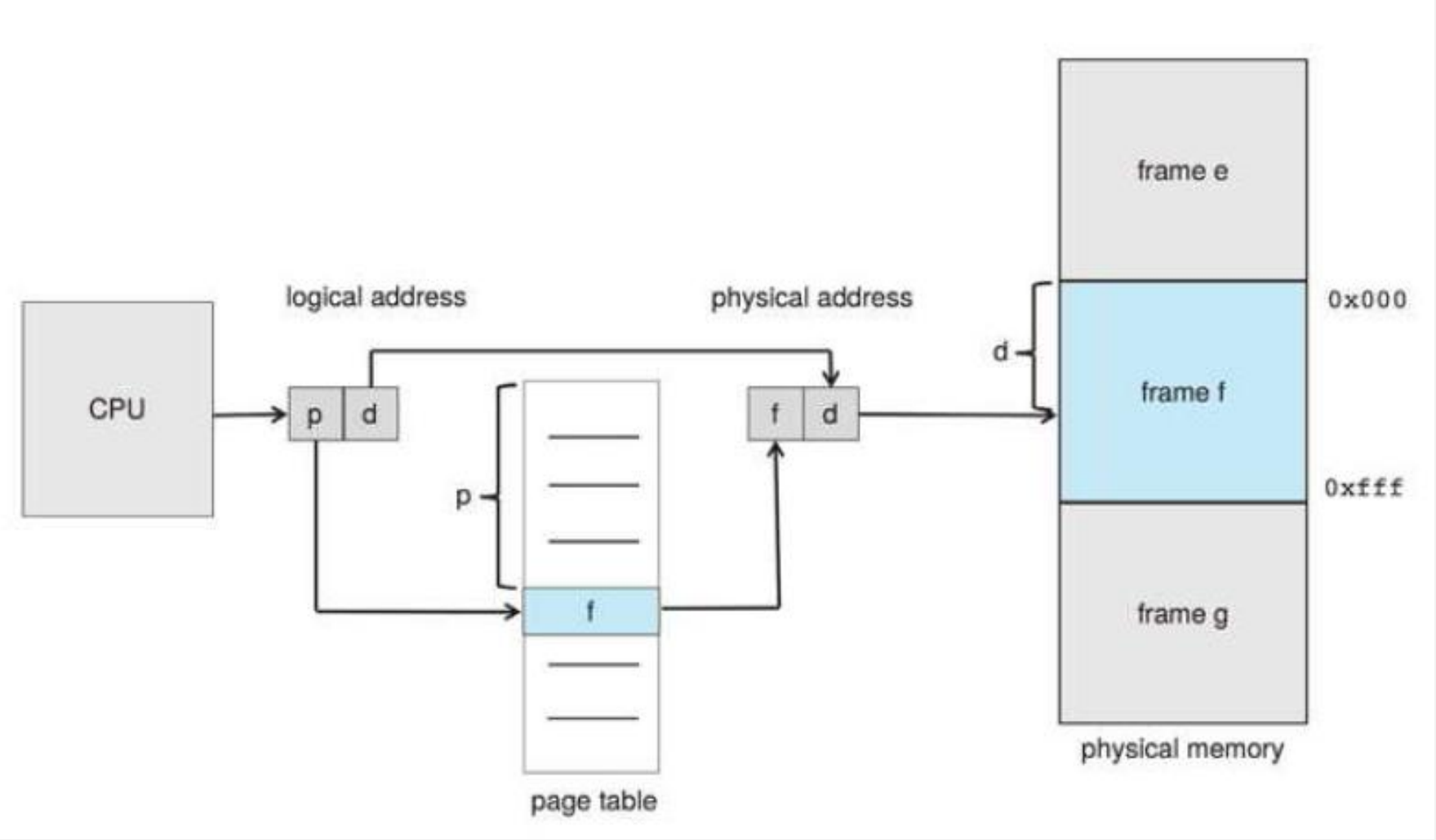
(f) Load process D

PAGINAÇÃO – CÁLCULO DO ENDEREÇO

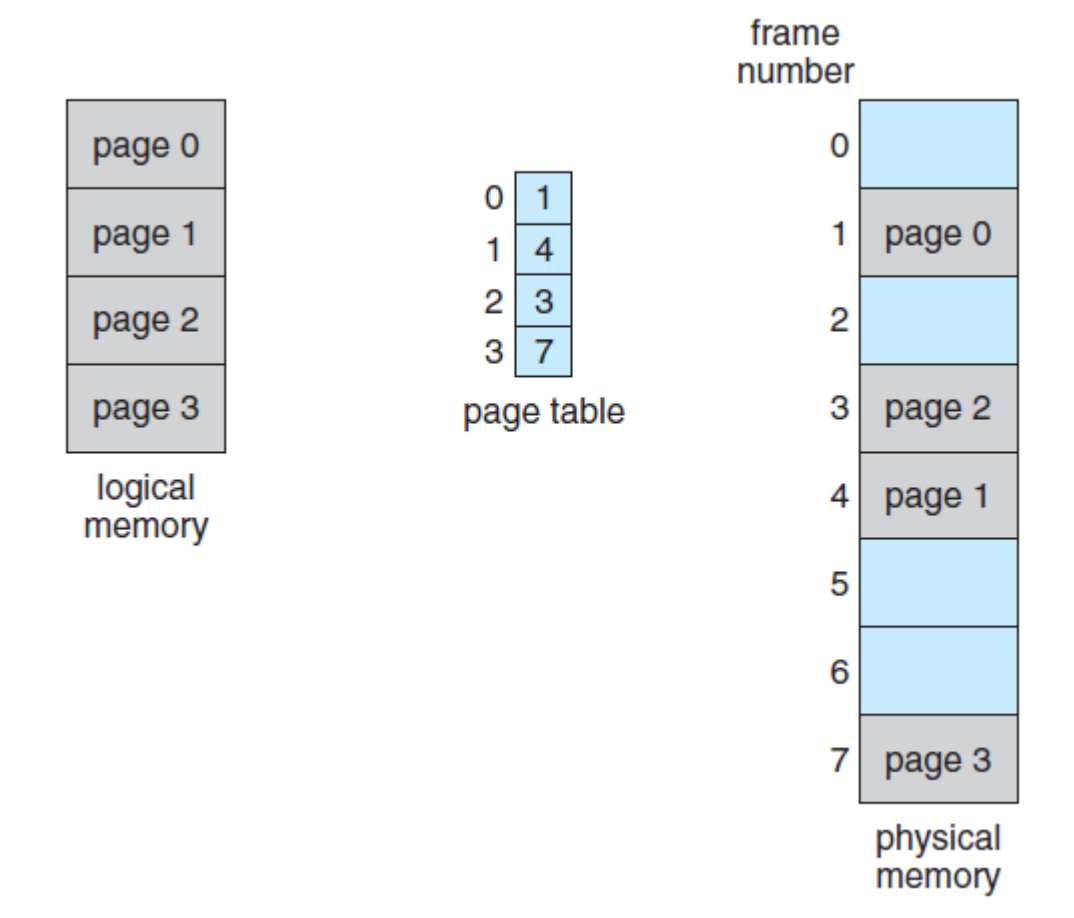
- **Passos** que o MMU utiliza para **traduzir** um **endereço lógico** gerado pelo CPU para um **endereço físico:**
  1. **Extrair o número da página p e usar como índice na tabela de páginas;**
  2. **Extrair o número do frame correspondente f da tabela de páginas;**
  3. **Substituir o número da página p no endereço lógico pelo número do frame f.**

Obs.: O offset d não varia, portanto, não é substituído.

PAGINAÇÃO – CÁLCULO DO ENDEREÇO



PAGINAÇÃO – CÁLCULO DO ENDEREÇO



PAGINAÇÃO – ENDEREÇAMENTO

- O **tamanho** da página (ou do frame) é definido pelo hardware
- Normalmente, é um **múltiplo de 2**, pois facilita a tradução dos endereços

- Espaço do endereçamento lógico → 2^m
- Tamanho de uma página → 2^n

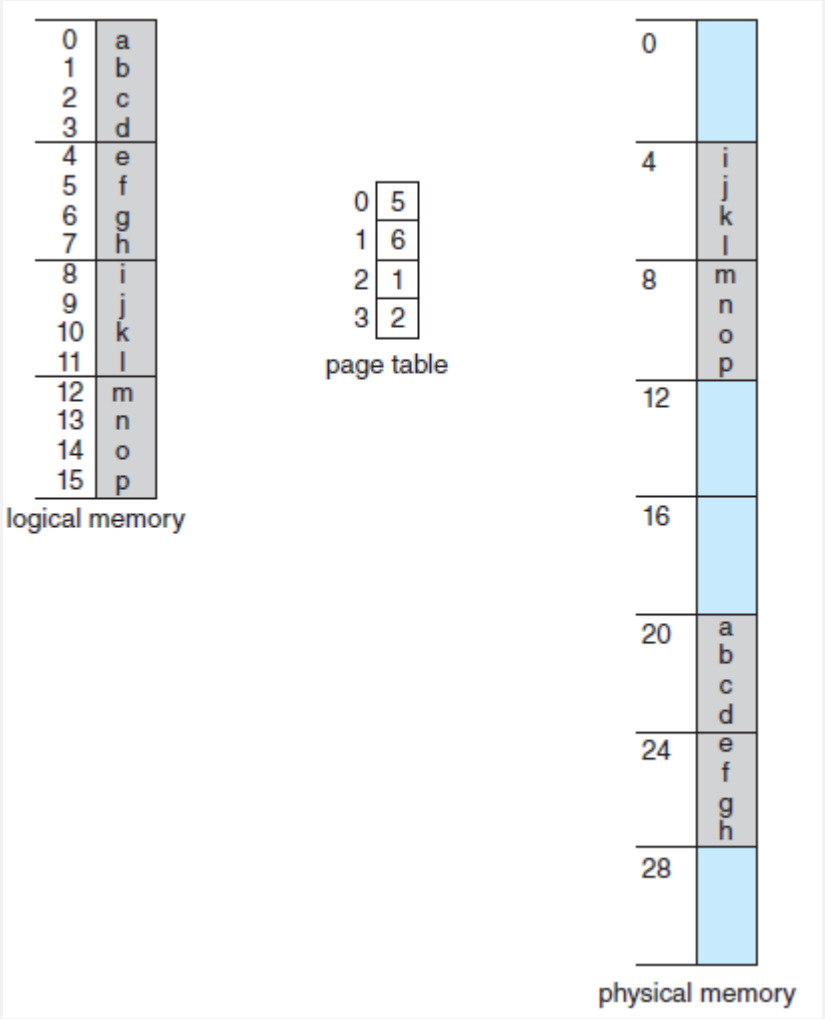
Número da página	Offset
m - n	n

PAGINAÇÃO – ENDEREÇAMENTO

- **Exemplo:**
- Endereço lógico → n = 2 e m = 4
- Tamanho de página de 4 bytes e tamanho da memória física de 32 bytes (8 páginas)

- Endereço lógico 0 é página 0 com offset 0
- Indexando a tabela de páginas, página 0 está no frame 5 → endereço físico 20

- Endereço lógico 4 é página 1 com offset 0
- Indexando a tabela de páginas, página 1 está no frame 6 → endereço físico 24



PAGINAÇÃO - COMENTÁRIOS

- **Paginação** é uma forma de realocação dinâmica
- Cada endereço lógico é relacionado pelo hardware a um endereço físico

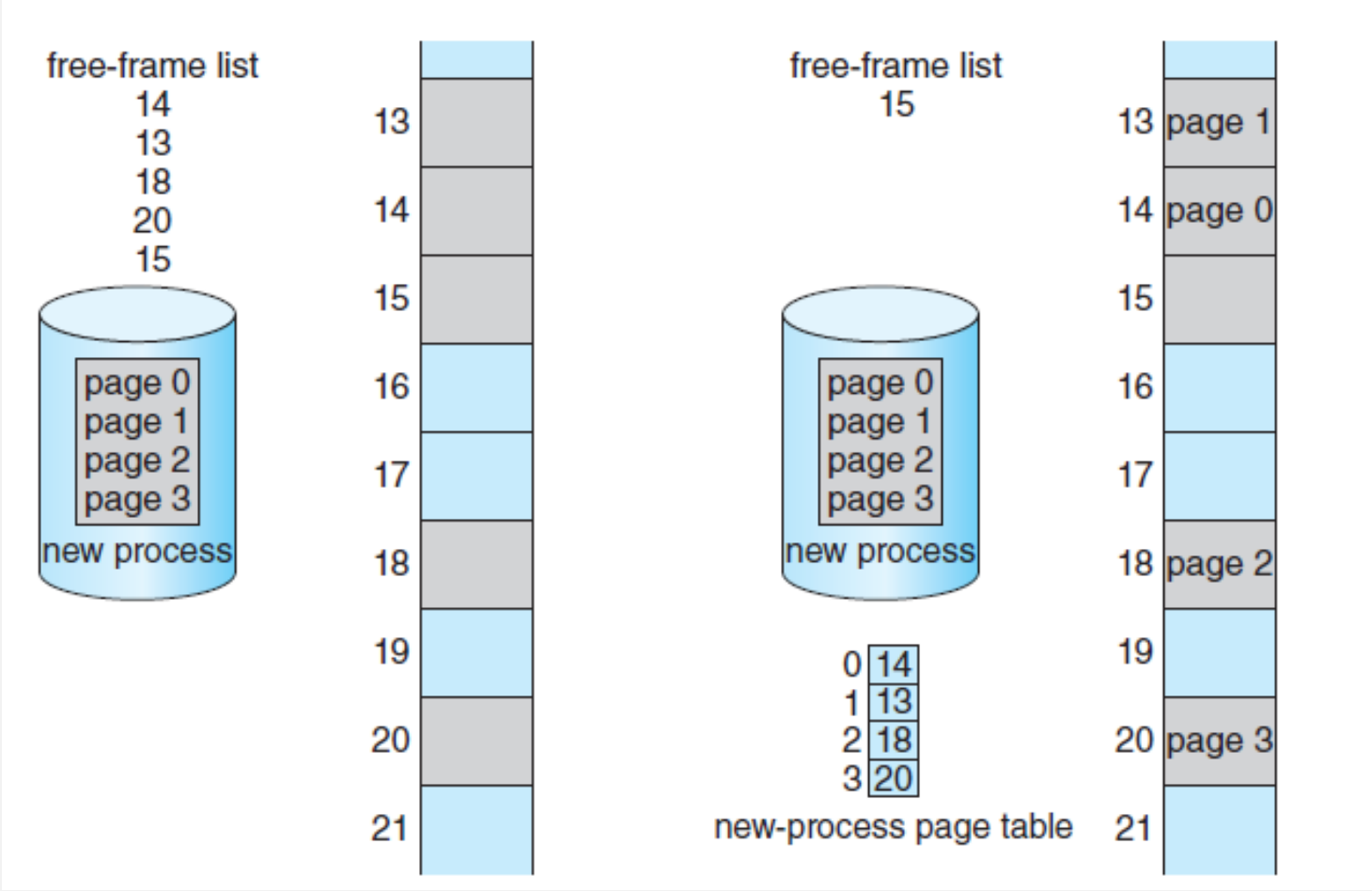
- Não acontece **fragmentação externa**: qualquer frame livre pode ser alocado a um processo (ou uma página de processo) que precisar.

- Pode acontecer **fragmentação interna**: frames são alocados como unidades; se o tamanho do processo não coincidir com o tamanho dos frames, algum frame pode não ser completamente preenchido.

PAGINAÇÃO
ALOCAÇÃO DE UM NOVO PROCESSO

- Processo novo para ser executado → tamanho expresso em páginas
- Cada página do processo necessita um frame
- SO guarda uma lista dos frames livres (tabela de frames)
- Se n frames estão disponíveis, estes são alocados ao novo processo
- Criada a tabela de páginas para referência

PAGINAÇÃO
ALOCAÇÃO DE UM NOVO PROCESSO



Antes da alocação

Depois da alocação

PAGINAÇÃO - CONCLUSÃO

- Um aspecto importante da paginação é a separação clara entre a visão da memória por parte do programador e a memória física real.

- **RESUMO:**
  - **Memória principal é dividida em frames e processos em páginas (mesmo tamanho)**
  - **Quando um processo é carregado para a memória, todas as suas páginas são carregadas nos frames disponíveis**
  - **Uma tabela de páginas é criada pelo SO para gerenciar os endereços**

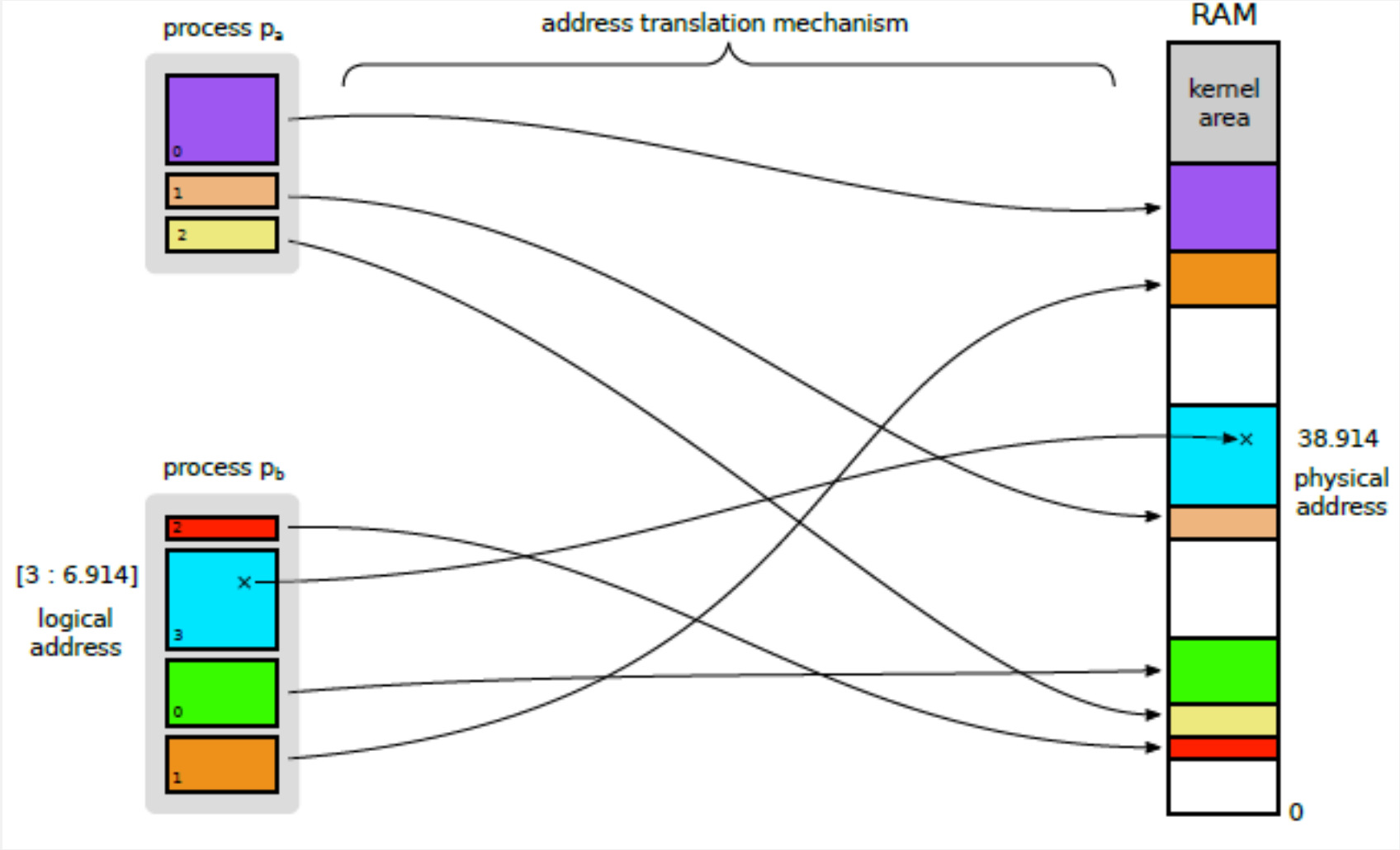
SEGMENTAÇÃO

- Outra técnica de gerenciamento de memória, parecida com paginação

- O **programa de usuário** é subdividido em **segmentos**
- Os segmentos **não precisam ser do mesmo tamanho**, no entanto, existe um **tamanho máximo** de segmento.
- Cada segment se comporta como uma partição de memória independente

- Mesma ideia da paginação, um endereço lógico utilizando segmentação consiste de duas partes: um **número de segmento** e um **offset** e criação de uma **tabela de segmentos.**

SEGMENTAÇÃO



SEGMENTAÇÃO

- Similar a particionamento dinâmico (ver aula anterior)

- **Diferença:** o espaço de endereçamento de cada processo não é mais visto como uma sequência linear de endereços lógicos, mas como uma coleção de áreas de tamanhos diversos e políticas de acesso distintas, denominadas segmentos.

- Elimina fragmentação interna
- No entanto, pode causar fragmentação externa

SEGMENTAÇÃO

- A paginação é invisível para o programador
- Segmentação é **visível para o programador** e fornecida como uma **conveniência** para organizar programas e dados

- **Programação modular**: programas e dados podem ser quebrados em vários segmentos
- **Limitação**: tamanho máximo do segmento

SEGMENTAÇÃO - CONCLUSÃO

- **RESUMO:**
  - Um processo é dividido em um número de segmentos que não precisam ser do mesmo tamanho
  - Quando um processo é carregado para a memória, todos os seus segmentos são alocados em regiões disponíveis da memória
  - Uma tabela de segmentos é criada pelo SO

  - **A segmentação possibilita um mecanismo "alto-nível" para o usuário**
  - **A paginação provê um mecanismo automático de alocação**

COMPARAÇÃO – ORGANIZAÇÃO DE MEMÓRIAS

- **Particionamento –** simplicidade e rapidez
- **Páginas –** oferece um grande espaço de endereçamento linear, enquanto elimina a fragmentação externa
- **Segmentos –** oferece múltiplos espaços de endereçamento para cada processo, oferecendo flexibilidade ao programador

CONCLUSÃO

- Apesar do processador Intel x86 oferecer as duas formas de organização de memória, a maioria dos sistemas operacionais que o suportam não fazem uso de todas as suas possibilidades:
- os sistemas da família Windows NT (2000, XP, Vista) e também os da família UNIX (Linux, FreeBSD) usam somente a organização por páginas.

- O antigo DOS e o Windows 3 usavam somente a organização por segmentos

PRÓXIMA AULA

- Memória virtual

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.