SISTEMAS OPERACIONAIS

AULA 12 – GERENCIAMENTO DE MEMÓRIA
CONCEITOS E DEFINIÇÕES – PARTE I

Prof.ª Sandra Cossul, Ma.

INTRODUÇÃO

- O CPU **compartilha** um conjunto de processos
- Como resultado do **escalonamento** destes processos, conseguimos obter **eficiência** na utilização do CPU e na velocidade de resposta do computador para os usuários
- Para isso, precisamos manter vários processos na memória, isto é, precisamos **compartilhar a memória**
- O compartilhamento dos processos na memória é uma função do SO e a essa tarefa chamamos de **gerenciamento da memória.**

GERENCIAMENTO DE MEMÓRIA

- Parte do SO que gerencia o uso da memória
- **Função:**
  - Gerenciar a memória de **modo eficiente:**
  - Mantendo o controle de quais partes da memória **estão em uso** e quais partes da memória **não estão**
  - **Alocando memória** aos processos quando eles precisam e **liberando-a** quando esses processos terminam

CONCEITOS INICIAIS

- **Memória lógica**
  - **memória que o processo enxerga**, ou seja, aquela que é capaz de endereçar e acessar usando as suas instruções
  - Endereços manipulados pelo processo são **endereços lógicos**
  - Cada processo possui a sua **memória lógica**, que é **independente** da memória lógica dos outros processos

- **Memória física**
  - Implementada pelos **circuitos integrados de memória** (e.g., hardware)
  - **Endereços físicos** são aqueles que vão para a memória física, ou seja, é usado para endereçar os circuitos integrados de memória

CONCEITOS INICIAIS

- **Espaço de endereçamento lógico**
  - Formado por todos os **endereços lógicos** que um processo pode gerar
  - Existe um **espaço** de endereçamento lógico **por processo**

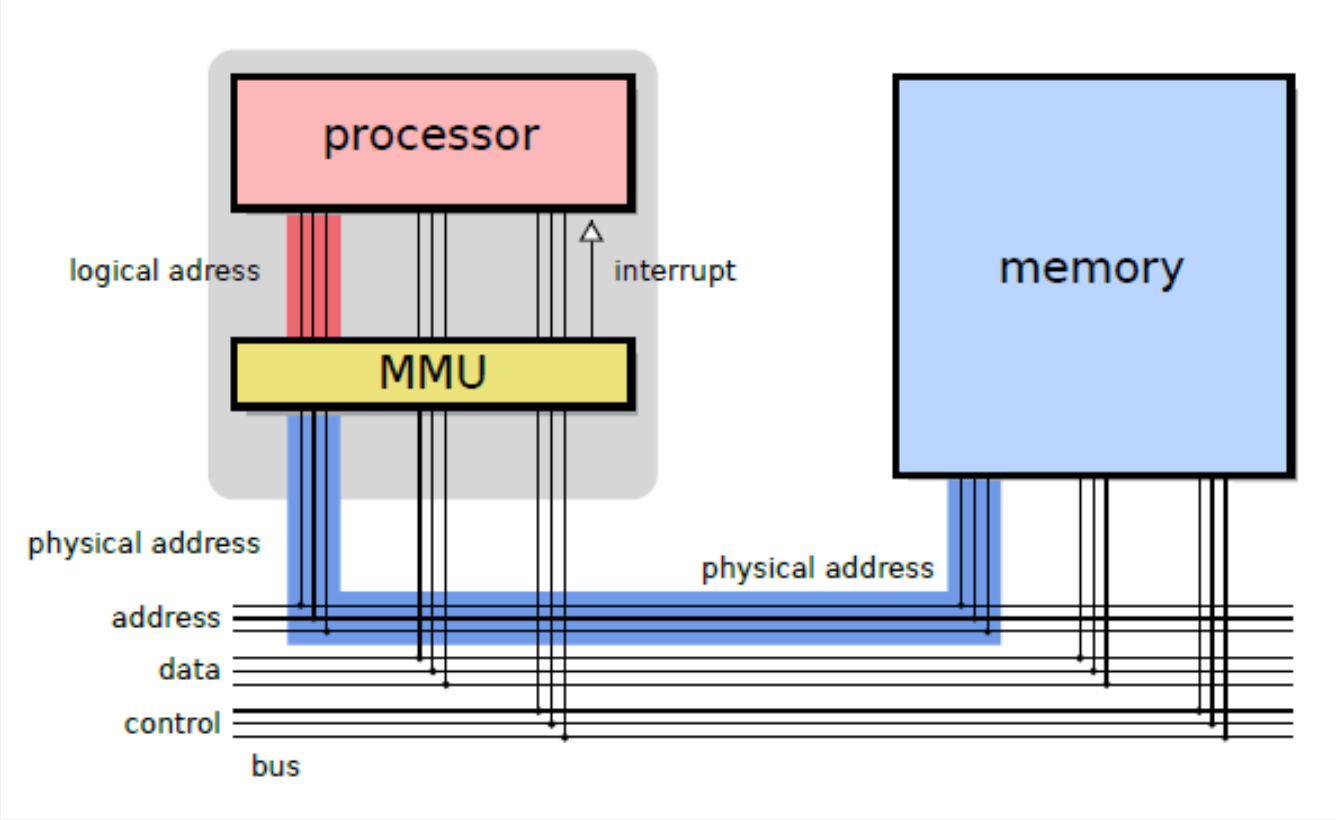
- **Espaço de endereçamento físico**
  - Formado por todos os **endereços aceitos** pelos **circuitos integrados de memória**

CONCEITOS INICIAIS

- **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 INICIAIS

- **Unidade de gerência de memória (MMU)**



MULTIPROGRAMAÇÃO X GERENCIAMENTO DE MEMÓRIA

- Um **gerenciamento** da memória **eficiente** é muito importante em **sistemas multiprogramados**

- A **memória** deve ser **alocada** para garantir um conjunto razoável de **processos prontos para executar** (nem muito poucos, nem processos demais) de forma que o processador fique **ocupado** (seja aproveitado todo o tempo de processador)

- Vamos discutir várias **técnicas de gerenciamento de memória**!

MULTIPROGRAMAÇÃO X GERENCIAMENTO DE MEMÓRIA

- **Multiprogragramação → vários processos na memória**
  - **Proteção –** Como proteger os processos uns dos outros?
  - **Necessidade de realocação –** Como tratar a realocação? O processo pode estar em diferentes posições da memória

- Todas as soluções envolvem o uso do CPU com a MMU.

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Realocação**
- **Proteção**
- **Compartilhamento**
- **Organização lógica**
- **Organização física**

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Realocação**
  - Memória é compartilhada
  - Programador não tem conhecimento de todos os processos que estão na memória principal quando em execução
  - Processos ativos são trocados (escalonados) – melhor uso do CPU
  - **Processos são alocados e realocados para áreas da memória disponíveis!**
  - O CPU e o SO devem **traduzir** as referência de memória encontrados no código do programa (endereços lógicos) para endereços de memória físicos, refletindo a **localização atual** do programa na **memória principal**

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Proteção**
  - processos devem ser protegidos de interferências de outros processos
  - programas em outros processos não devem ser capazes de referenciar posições de memória (escrita ou leitura) sem permissão.
  - **Todas as referências de memória geradas por um processo devem ser verificadas em tempo de execução para garantir que elas referenciam apenas o espaço de memória alocado para esse processo.**
    - Função do hardware (CPU) e não do SO

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Compartilhamento**
  - Qualquer mecanismo de proteção deve ter **flexibilidade** para permitir que vários processos acessem a mesma área de memória
    - Um programa com vários processos --> compartilhamento da mesma cópia do programa
    - Processos cooperando em uma tarefa → compartilhamento da mesma estrutura de dados

  - **Sistema de gerenciamento de memória deve permitir acesso controlado a áreas compartilhadas de memória sem comprometer a proteção!**

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

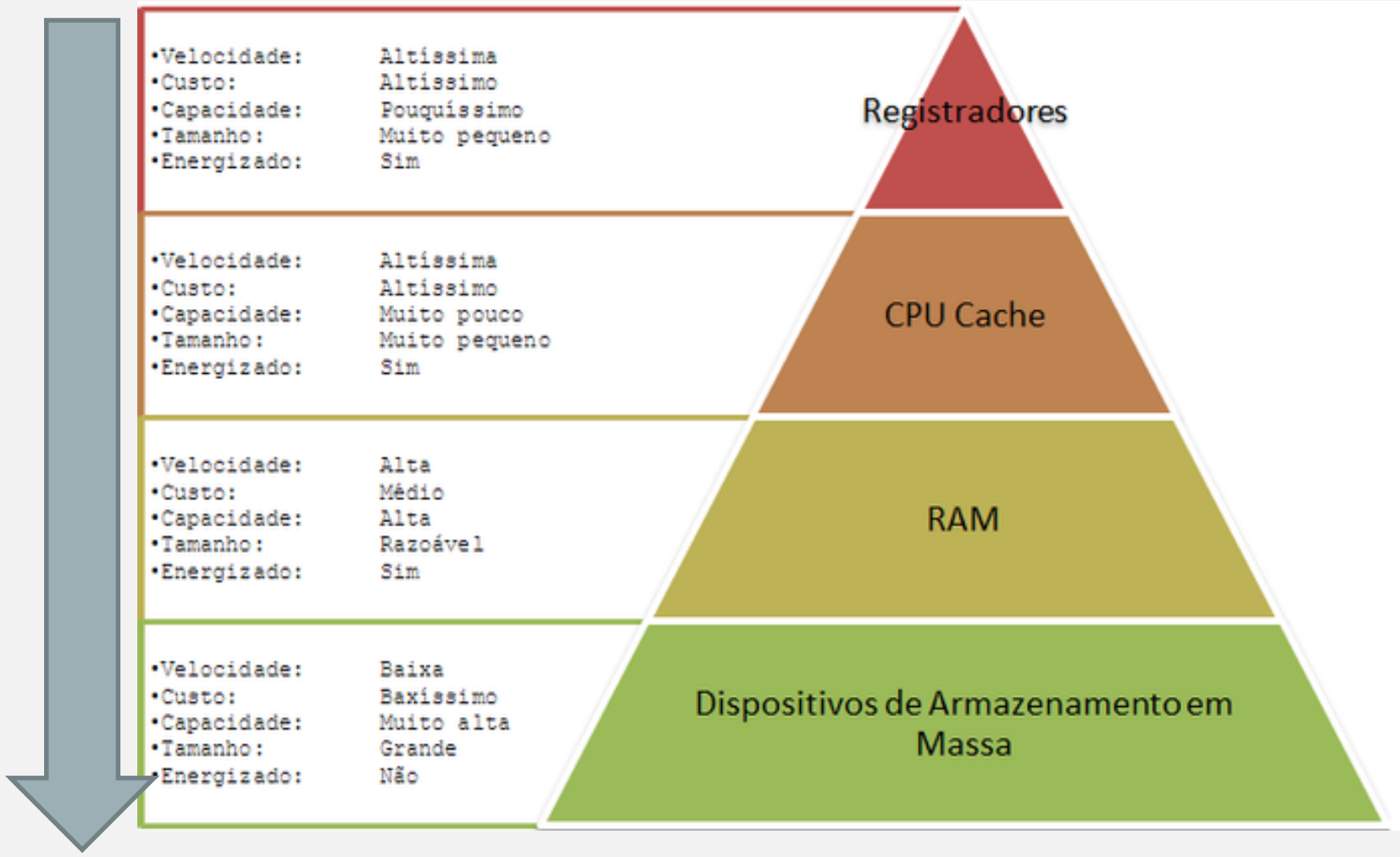
- **Organização lógica**
  - O CPU e o SO entendem **programas e dados** na forma de **módulos,** o que apresenta algumas vantagens:
    - módulos podem ser escritos e compilados independentemente
    - diferentes graus de proteção (apenas leitura, apenas execução) podem ser atribuídos a diferentes módulos
    - possibilidade de utilizar mecanismos onde módulos podem ser compartilhados entre processos

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Organização física**
  - A memória é organizada em **níveis**
    - **Memória secundária (HD, SSD)** – armazenamento permanente de programas e dados
    - **Memória principal (RAM) –** armazena temporariamente programas e dados em uso
  - O fluxo de informações entre memória secundária e principal deve ser gerenciado pelo SO.

REQUERIMENTOS DO GERENCIAMENTO DE MEMÓRIA

- **Diminuição do custo por bit**
- **Aumento da capacidade**
- **Aumento do tempo de acesso**
- **Diminuição da frequência de acesso à memória pelo processador**



GERENCIAMENTO DE MEMÓRIA

- **Operação principal do gerenciamento de memória** → trazer processos da memória secundária para a memória principal para serem executados pelo processador!

TROCA DE PROCESSOS NA MEMÓRIA (*SWAPPING*)

- **Problema:** o CPU é muito mais rápido que a E/S, e mesmo utilizando multiprogramação, pode acontecer de ter todos os processos na memória esperando por E/S e o CPU ficar ocioso.

- **Solução:** troca de processos na memória principal
  - Fila de longo prazo de requisições de processos na memória (trazidos um por vez)
  - Surge a situação que nenhum dos processos na memória estará pronto
  - Em vez de ficar ocioso, o CPU troca (*swap*) um desses processos para uma fila intermediária
  - SO traz outro processo da memória para continuar a execução

- **Desvantagem:** a troca de processo é uma operação de E/S → custosa

PARTICIONAMENTO DA MEMÓRIA

- Uma das formas mais simples de organização da memória e tradução de endereços lógicos em físicos consiste em **dividir a memória física em N partições**
- Em cada partição da memória física é carregado um processo!

- **Particionamento fixo**
  - **Tamanho das partições:** fixas ou variáveis

- **Particionamento dinâmico**

PARTICIONAMENTO DA MEMÓRIA (FIXO)

- Dividir a memória disponível em partes (partições) de **tamanho fixo.**

- Processos com **tamanho menor ou igual ao tamanho da partição** são trazidos para a memória e colocados em uma partição **disponível**

- Se as partições estão cheias, os processos são trocados

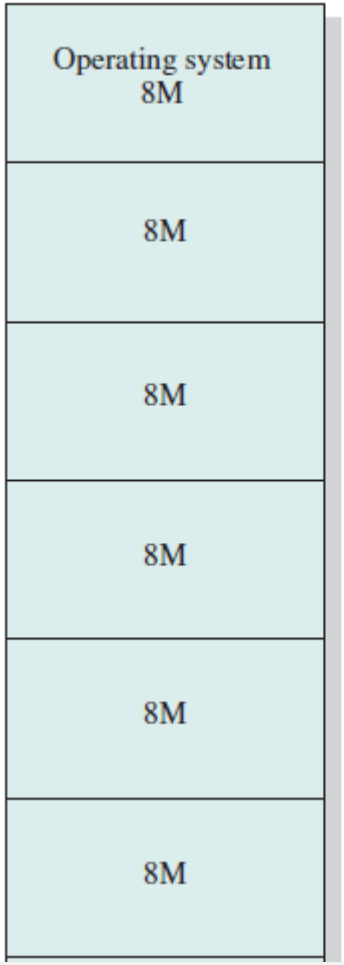
PARTICIONAMENTO DA MEMÓRIA

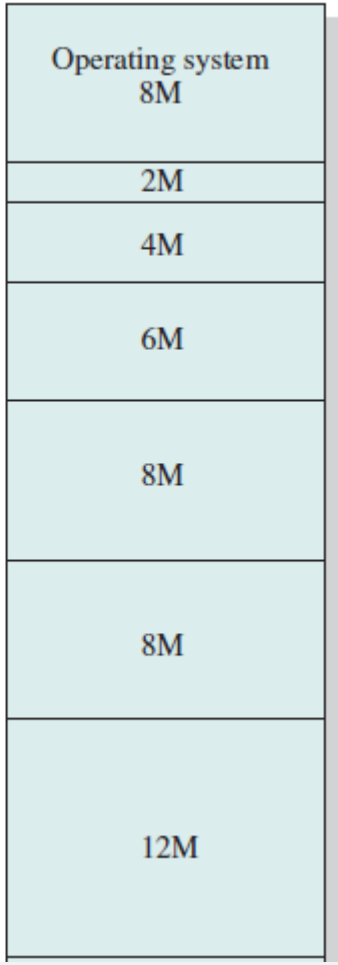
- **Problemas:**
  - um programa pode ser muito grande e não caber em uma partição
  - uso ineficiente da memória (programas pequenos ocupam toda uma partição) – **fragmentação interna**

- **Solução**
  - Utilizar partições de tamanho variável
  - O processo recebe a quantidade de memória exigida

PARTICIONAMENTO DA MEMÓRIA

Particionamento fixo





Particionamento variável

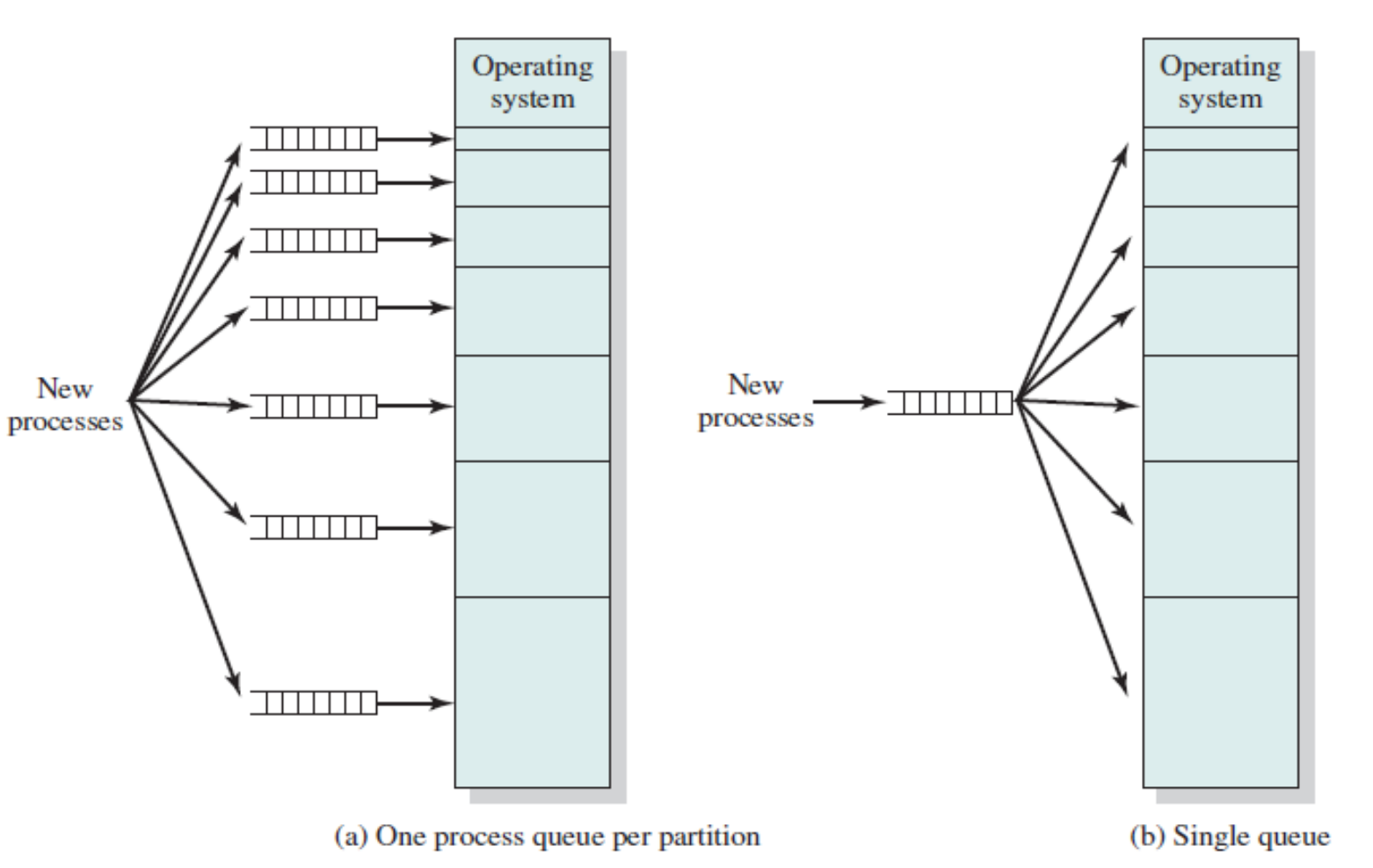
ALGORITMOS DE ATRIBUIÇÃO DOS PROCESSOS ÀS PARTIÇÕES

- **Para partições de tamanho fixo:**
  - não precisa ter "preocupação" em relação ao algoritmo de alocação das partições já que todas tem o mesmo tamanho.
  - Processos que não estão prontos para executar são substituídos por processos prontos
  - A escolha de qual processo será substituído já foi estudado nos algoritmos de escalonamento.

ALGORITMOS DE ATRIBUIÇÃO DOS PROCESSOS ÀS PARTIÇÕES

- **Para partições de tamanho variável:**
  - **Opção 1** → designar cada processo para a menor partição disponível na qual o processo vai "servir"
    - necessária uma fila de escalonamento para cada partição manter processos trocados destinados a essa partição
  - **Opção 2** → utilizar uma única fila para todas as partições, de forma que as partições sejam "melhor" ocupadas

ALGORITMOS DE ATRIBUIÇÃO DOS PROCESSOS ÀS PARTIÇÕES



PARTICIONAMENTO DA MEMÓRIA

- **Partições de tamanho variável**
  - **Vantagens:** melhor uso da memória (minimização da fragmentação interna) e implementação simples
  - **Desvantagens:**
    - Número de partições especificadas limitam o número de processos ativos no sistema
    - Como o tamanho das partições é pré-definido, processos pequenos vão utilizar partições de forma ineficiente.
  - A técnica de particionamento não é mais utilizada nos SOs modernos.

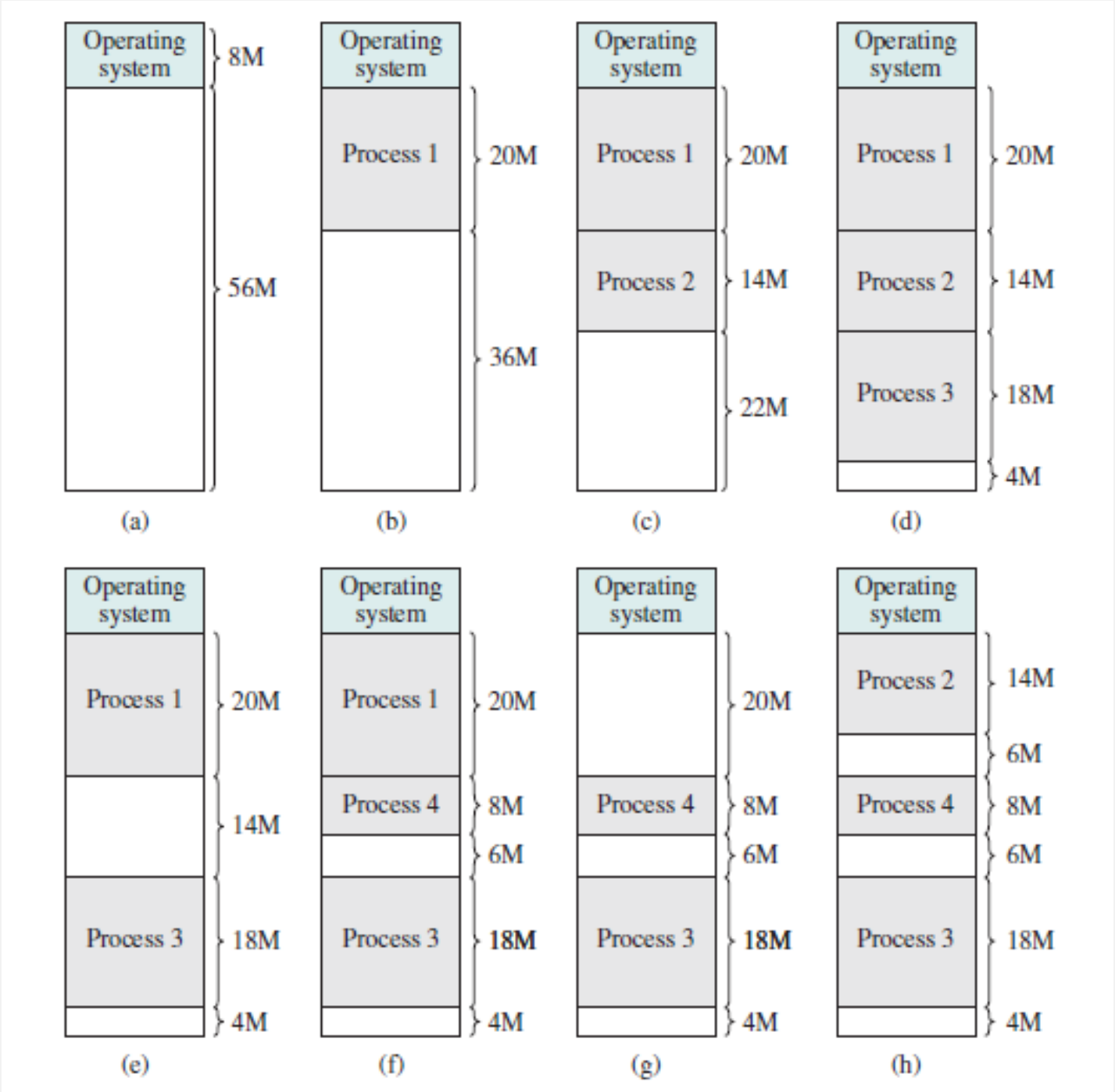
PARTICIONAMENTO DA MEMÓRIA

- **Particionamento fixo**
  - **Tamanho das partições:** fixas ou variáveis
- **Particionamento dinâmico**

PARTICIONAMENTO DA MEMÓRIA

- **Partições dinâmicas**
  - partições tem tamanho e número variado
  - quando um processo é trazido para a memória principal, é alocada exatamente a quantidade de memória que ele precisa.

Particionamento dinâmico da memória



PARTICIONAMENTO DA MEMÓRIA

- **Partições dinâmicas**
  - **Problema:**
    - memória vai ficando com "buracos", ou seja, ocorre a fragmentação da memória e o uso da memória decai
    - **fragmentação externa –** memória que é externa a todas as partições vai se tornando cada vez mais fragmentada.

PARTICIONAMENTO DA MEMÓRIA

- **Partições dinâmicas**

  - **Solução para fragmentação externa: compactação**
    - de tempos em tempos, o SO move os processos tornando-os contíguos e a memória "livre" fica em único bloco novamente.
    - processo custoso (usa tempo de processamento)
    - Compactar → necessidade de realocação → deve ser feito sem invalidar as referências de memória do programa

ALGORITMOS DE ATRIBUIÇÃO DOS PROCESSOS

- **O SO deve decidir como melhor designar os processos para a memória** (como melhor preencher os blocos livres de memória)
  - **Ideia**: evitar executar compactação
  - **Objetivo dos algoritmos:** escolher dentre os blocos de memória disponíveis que são iguais ou maiores que o processo que vai ser trazido.
  - **Como escolher? Algoritmos:**
    - Best-fit
    - First-fit
    - Next-fit

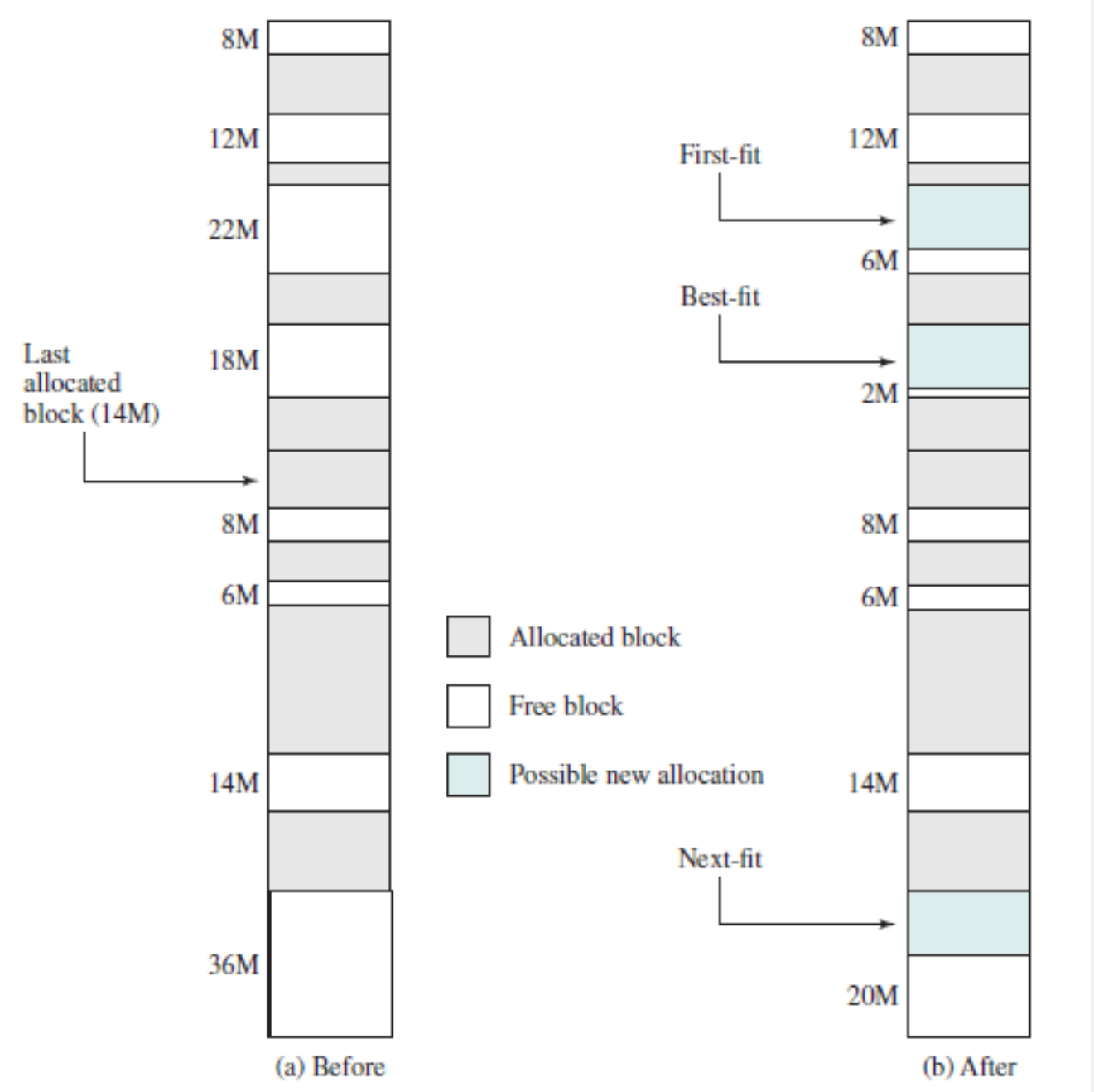
ALGORITMOS DE ATRIBUIÇÃO DOS PROCESSOS

- **Best-fit –** escolhe o bloco de tamanho mais próximo ao requisitado
- **First-fit –** "varre" a memória desde o início e escolhe o primeiro bloco disponível que tem tamanho suficiente
- **Next-fit –** "varre" a memória a partir do último local de atribuição e escolhe o próximo bloco disponível que tem tamanho suficiente

- A escolha do melhor algoritmo vai depender da sequência de troca dos processos e do tamanho dos processos.

Algoritmos de alocação de memória

→ **alocar um processo com 16MB**



REALOCAÇÃO DE PROCESSOS

- **Partições fixas de tamanho variável com filas separadas** → processo vai ser alocado sempre na mesma partição
- **Partições fixas de tamanho fixo / partições fixas de tamanho variável com uma única fila / partições dinâmicas / compactação** → processo pode ser alocado em diferentes partições devido a troca de processos na memória (swapping)

  –

- **Problema:** as posições de memória (instruções e dados) referenciados por um processo não são fixas (podem ser trocadas ao longo da execução de um processo)
- **Solução:** fazer uma distinção dos tipos de endereços!

REALOCAÇÃO DE PROCESSOS

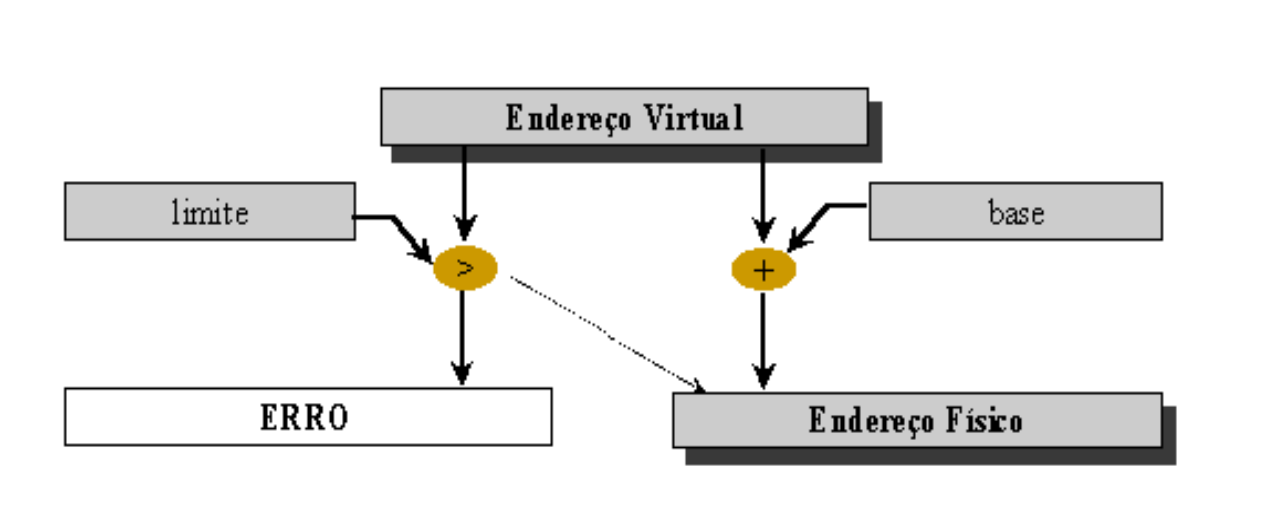
- **Endereço lógico –** referência para uma posição de memória independente; uma tradução deve ser feita para um endereço físico antes que a posição de memória seja acessada.
- **Endereço físico –** ou endereço absoluto, localização real na memória principal.
- **Endereço relativo –** endereço lógico expresso relativo a uma posição conhecida, normalmente um valor guardado num registrador do CPU

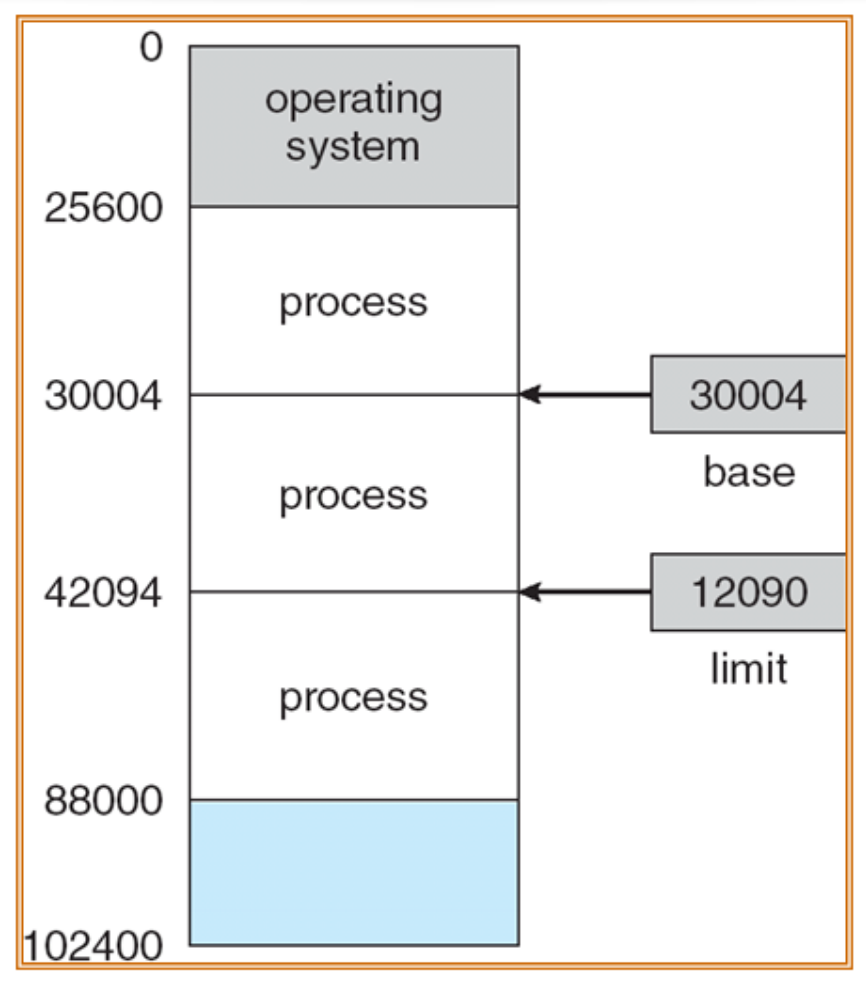
- **Em tempo de execução, é necessário "traduzir" os endereços relativos para endereços absolutos (físicos)!**

REALOCAÇÃO DE PROCESSOS

- **2 registradores - base e limite**
- 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**
- O registrador-base torna impossível a um processo acessar a qualquer parte de memória abaixo de si mesmo.
- Automaticamente, a MMU adiciona o conteúdo do registrador-base a cada endereço de memória gerado.
- Endereços são comparados com o registrador-limite para prevenir acessos indevidos.

REALOCAÇÃO DE PROCESSOS





REALOCAÇÃO DE PROCESSOS

- **Proteção: cada processo fica isolado dentro dos valores dos registradores base e limite, ficando protegido de acesso por outros processos.**

- O CPU compara cada endereço gerado com os registradores, prevenindo acesso indevido à memória

- Se o endereço estiver fora do limite especificado, é gerado um **erro de acesso ilegal da memória.**

EXERCÍCIO 1

- Considere um sistema com processos alocados de forma contígua na memória.
- Em um dado instante, a memória RAM possui os seguintes "buracos", em sequência e isolados entre si: 5K, 4K, 20K, 18K, 7K, 9K, 12K e 15K.
- Indique a situação final de cada "buraco" de memória após a seguinte sequência de alocações: 12K → 10K → 5K → 8K → 10K.
- Considere as estratégias de alocação *first-fit*, *best-fit* e *next-fit*.

EXERCÍCIO 2

- Considerando partições de memória de 100k, 500k, 200k, 300k, 600k (nessa ordem), como cada um dos algoritmos abaixo alocaria os seguintes processos: 212k, 417k, 112k, 426k (nessa ordem)?
- Qual algoritmo foi mais eficiente?
  - First-fit
  - Best-fit
  - Next-fit

PRÓXIMA AULA

- Outros mecanismos de gerenciamento de memória:
  - Paginação
  - Segmentação

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.