Capítulo

9

Organização e gerenciamento da memória real

Sumário

PEARSON Prentice Hall

Objetivos

- **Este capítulo apresenta:**
  - A necessidade do gerenciamento da memória real (também denominada física).
  - A hierarquia da memória.
  - Alocação de memória contígua e não contígua.
  - Multiprogramação por partição fixa e variável.
  - Troca (swapping) de memória.
  - Gerenciamento de memória com mapas de bit

PEARSON Prentice Hall

9.1 Introdução

- **A memória divide-se em camadas:**

* **Cache**
  - Pequena quantidade
    * k bytes
  - Alto custo por byte
  - Muito rápida
  - Volátil

* **Memória Principal**
  - Quantidade intermediária
    * M bytes
  - Custo médio por byte
  - Velocidade média
  - Volátil

* **Disco**
  - Grande quantidade –
    * G bytes
  - Baixo custo por byte
  - Lenta
  - Não volátil

9.2 Organização da memória

- **A memória pode ser organizada de diferentes formas:**

  Um processo usa todo o espaço de memória.
  Cada processo obtém uma partição exclusão na memória.
  Alocada dinâmica ou estaticamente.

- **Tendência: a necessidade de memória das aplicações tende a aumentar ao longo do tempo para atender à capacidade de memória principal.**

9.3 Gerenciamento de memória

- **Estratégias para obter um desempenho de memória ideal:**

Gerenciamento executado por um gerenciador de memória:

Garante isolamento mútuo entre processos (proteção);

Mantém informação das áreas de memória em uso;

Aloca memória RAM para novos processos (fork());

Faz o swapping transparente entre memória principal e disco;

Atende a requisições de incremento de espaço de memória;

Mantém o mapeamento de memória virtual para memoria física;

Implementa a política de alocação de memória para os processos.

9.4 Hierarquia de memória

- **Memória principal**

  Deve armazenar apenas instruções de programa necessárias no momento e dados.

- **Armazenamento secundário**

  Armazena dados e programas que não são constantemente necessários.

- **Memória cache**

  Sua velocidade é extremamente alta.

  Normalmente se localiza no próprio processador.

  Os dados mais comumente usados são copiados para o cache para que possam ser acessados mais rapidamente.

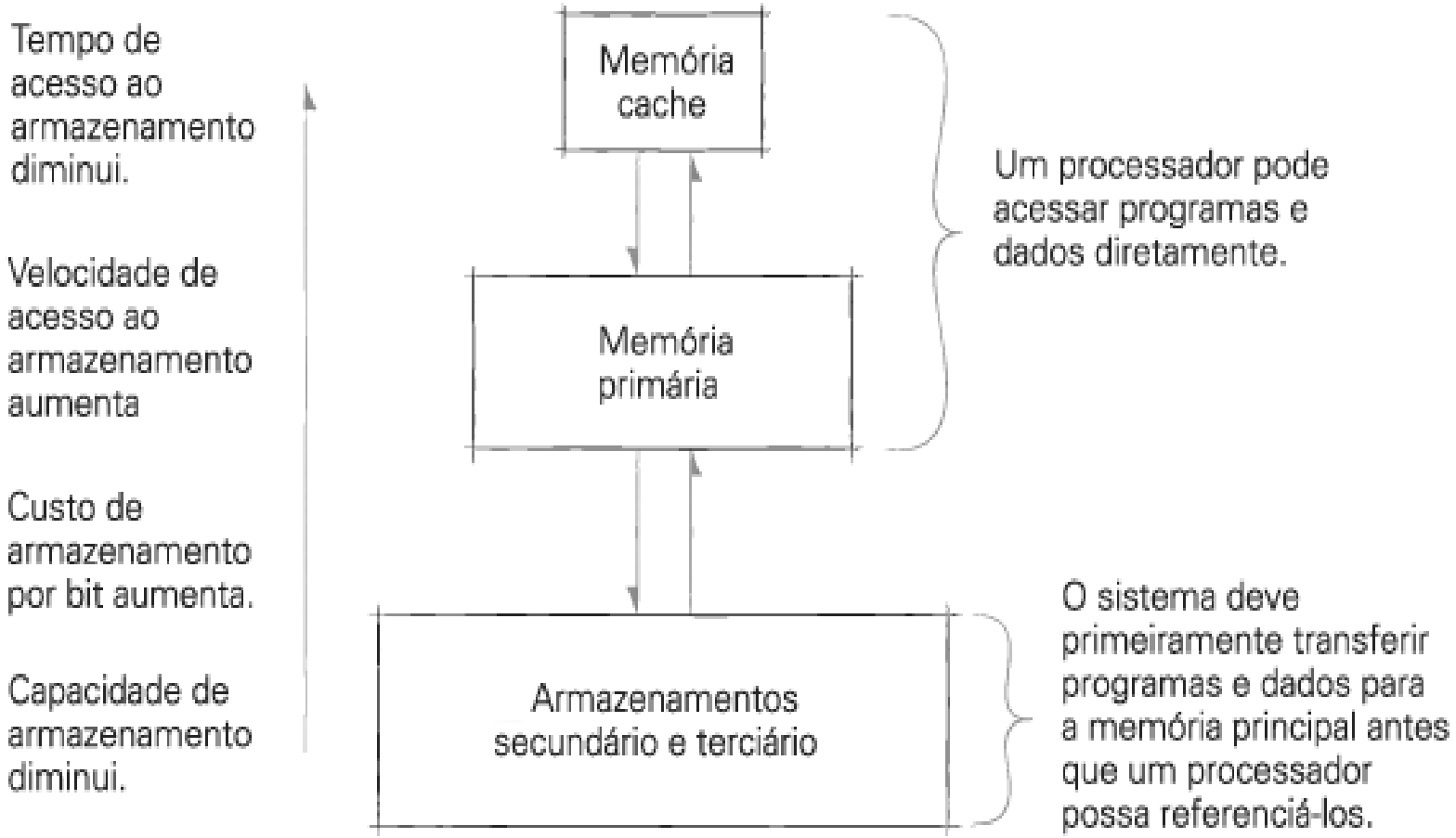
  Uma pequena quantidade de cache é suficiente para melhorar o desempenho.

  Isso se deve à localidade temporal.

PEARSON Prentice Hall

9.4 Hierarquia de memória

Figura 9.2 Organização da hierarquia da memória.



9.5 Estratégias de gerenciamento de memória

As estratégias dividem-se em diversas categorias:

Estratégias de busca

Sob demanda ou antecipada.

Determinam que porção de dados será transferida em seguida.

Estratégias de posicionamento

Determinam em que lugar serão colocados na memória principal os dados que estão chegando.

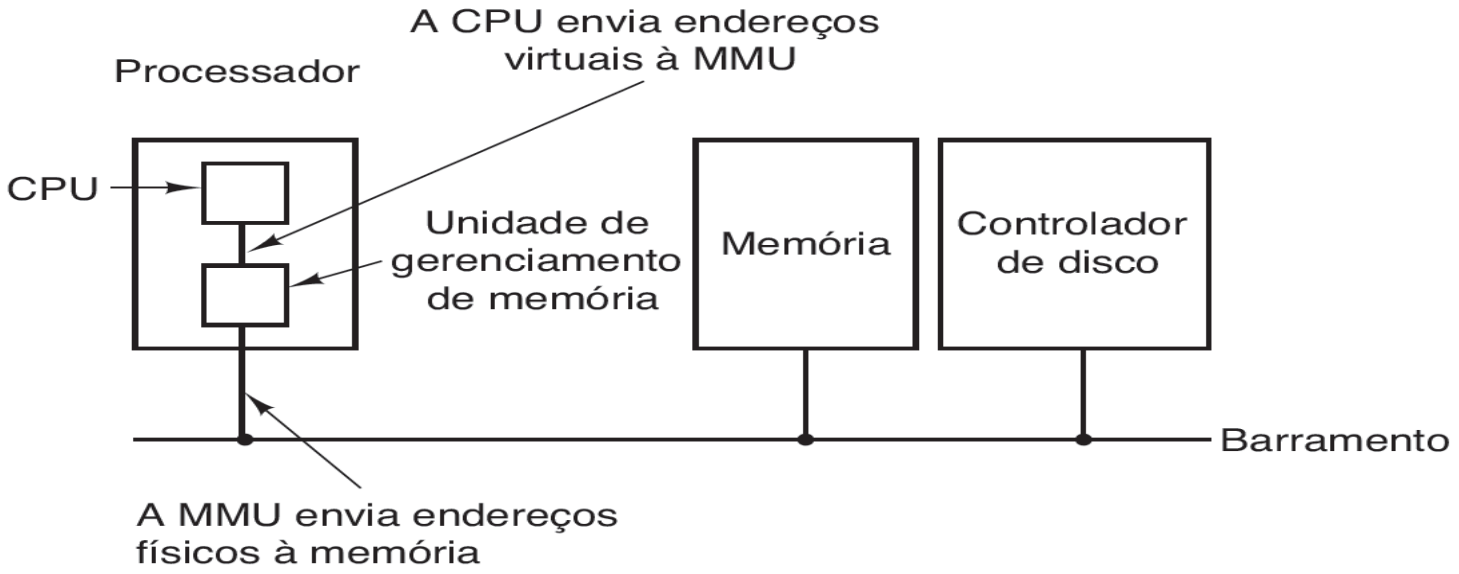
Estratégias de substituição

Determina que dados serão removidos da memória principal para liberar espaço.

9.5.1 Tradução de endereços

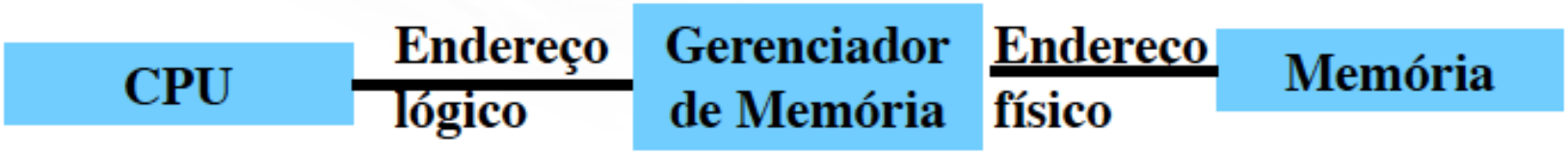
- Uma das funções mais importantes do sistema de **gerenciamento de memória** é a **tradução** dos **endereços lógicos** utilizados pelos processos para **endereços físicos** reais.

- A MMU (**Memory-Managment Unit**) é o modulo hardware responsável por realizar a conversão entre os endereços lógicos e físicos



9.5.1 Tradução de endereços

- **Memória Lógica -** É aquela que o processo enxerga, o processo é capaz de acessar.
- **Memória Física -** É aquela implementada pelos circuitos integrados de memória, pela eletrônica do computador (memória real)



- **Funções MMU:**
  - Verificar a posibilidade de conversão.
  - Comprovar as permissões de acesso do processo ao endereço de memória.
  - Converter endereços lógicos em físicos.
- O mecanismo de tradução da MMU depende do sistema de gerência de memória utilizado e vice-versa.

9.5.2 Relocação

- Capacidade de **mover** um programa de uma região da memória principal para uma outra **sem invalidar** as referências de memória dentro do programa;
- O **programador não deve se preocupar** com o local onde o programa (processo) será carregado para execução.
- Durante a execução, o processo poderá **sair** da memória e **retornar** para um local diferente.
- Referências devem ser resolvidas para **endereços de memória física**.
- O hardware do processador e o SO devem ser capazes de traduzir os endereços de referência de memória no código do programa para o endereço físico da memória

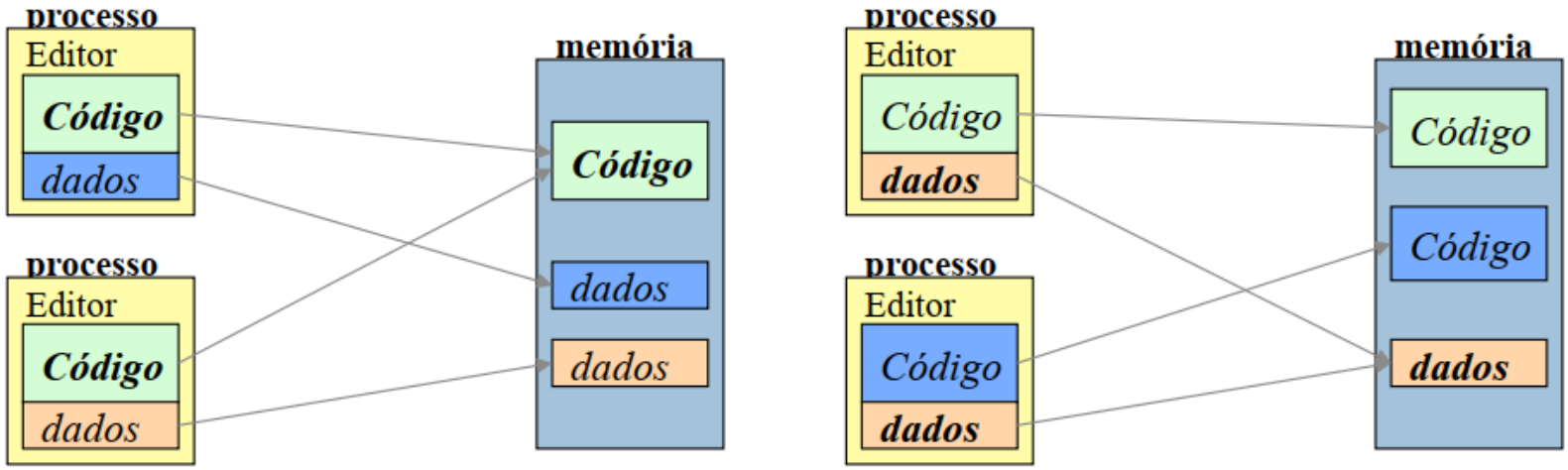
PEARSON
Prentice Hall

9.5.3 Proteção

- Proteção
  - O sistema de gerência de memória deve assegurar a proteção do **código** e dos **dados** dos processos contra acessos **acidentais ou intencionados** de outros processos.
  - Também deve proteger o **código e dados do S.O**.
  - Processos **não** devem poder referenciar posições de memória em outros processos sem permissão prévia.

9.5.4 Compartilhamento

- Existem grandes áreas dos programas (código e dados) que podem ser **compartilhados** por diferentes processos para reduzir os requisitos de memória e aumentar a multi-programação do sistema.
  - Deve-se permitir que vários processos acessem a mesma área de memória principal.
  - Deve-se permitir o compartilhamento sem **comprometer** o requisito de **proteção**.



9.6 Alocação de memória contígua e não contígua

Formas de organização dos programas na memória:

Alocação contígua

- O programa deve estar em um bloco único de endereços contíguos.
- Às vezes é impossível encontrar um bloco grande o suficiente.
- Sua sobrecarga é baixa.

Alocação não contígua

- O programa é dividido em porções denominadas segmentos.
- Os segmentos podem ser posicionados em partes diferentes da memória.
- É fácil encontrar "lacunas" nas quais o segmento possa se encaixar.
- Pelo fato de poder haver mais processos simultâneos na memória, isso compensa a sobrecarga própria dessa técnica.

PEARSON Prentice Hall

9.7 Alocação de memória contígua em sistema monousuário

- **Um usuário detinha o controle de toda a máquina**
  - Os recursos não tinham de ser compartilhados.
  - Originalmente, não havia nenhum sistema operacional no computador.
    - O programador escrevia o código para executar o gerenciamento de recursos.
  - Sistema de controle de entrada/saída (IOCS)
    - Bibliotecas de códigos pré-escritos para gerenciar dispositivos de E/S.
    - Precursor dos sistemas operacionais.

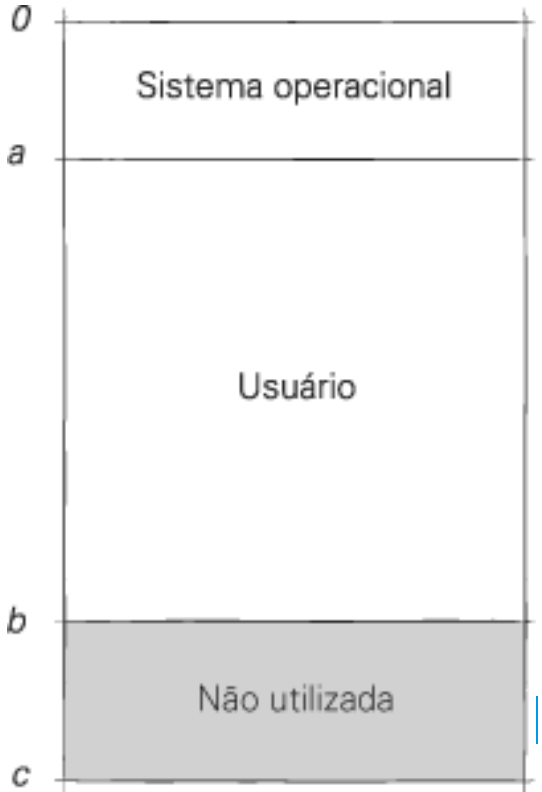
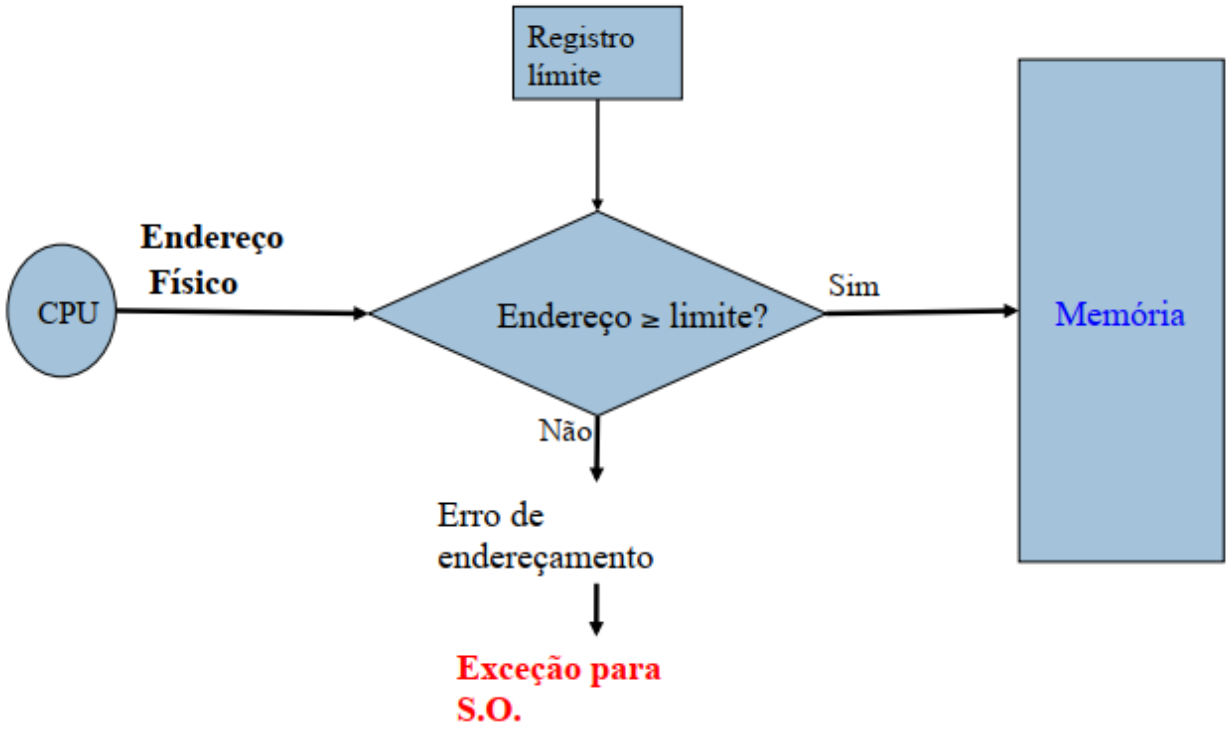


Figura 9.3 Alocação de memória contígua em sistema monousuário.

9.7 Alocação de memória contígua em sistema monousuário



- E se o tamanho do programa for maior do que a memória disponível?
  - Neste caso utiliza-se a técnica conhecida como **overlay**:
    - O programa é dividido em módulos que são executados independentemente na mesma área de memória

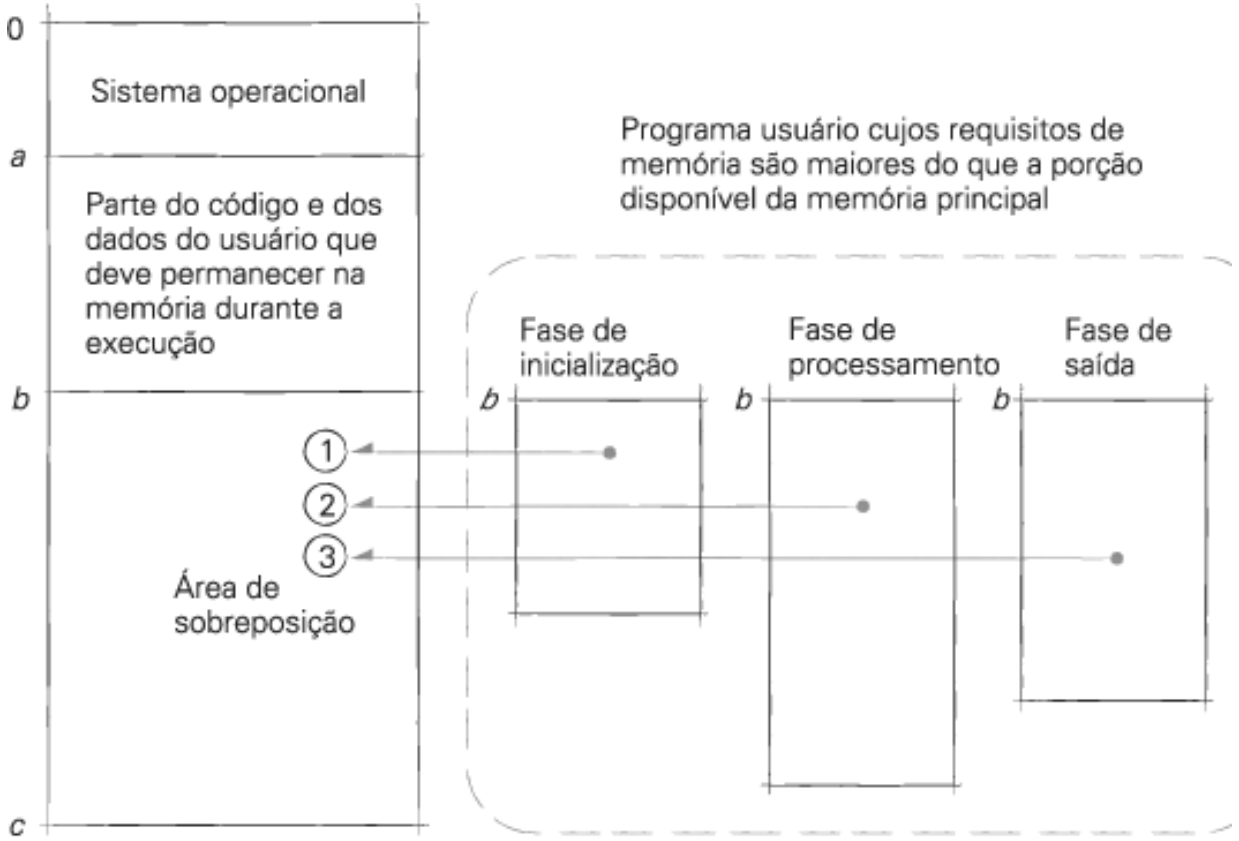
9.7.1 Sobreposições (Overlays)

- **Sobreposição: técnica de programação destinada a superar as limitações da alocação contígua.**
  - O programa é dividido em seções lógicas.
  - Coloca na memória apenas as seções ativas no momento.
  - **Apresenta sérios inconvenientes:**
    - Tem dificuldade de organizar as sobreposições a fim de usar a memória principal de modo eficaz.
    - Dificulta alterações no programa.
  - **A memória virtual cumpre objetivos semelhantes.**
    - Como o IOCS, a memória virtual protege os programadores contra problemas complicados, como é o caso do gerenciamento de memória.

PEARSON Prentice Hall

9.7.1 Sobreposições (Overlays)

Figura 9.4 Estrutura de sobreposição.



1 Carregue a fase de inicialização em *b* e execute.

2 Depois carregue a fase de processamento em *b* e execute.

3 Então carregue a fase de saída em *b* e execute.

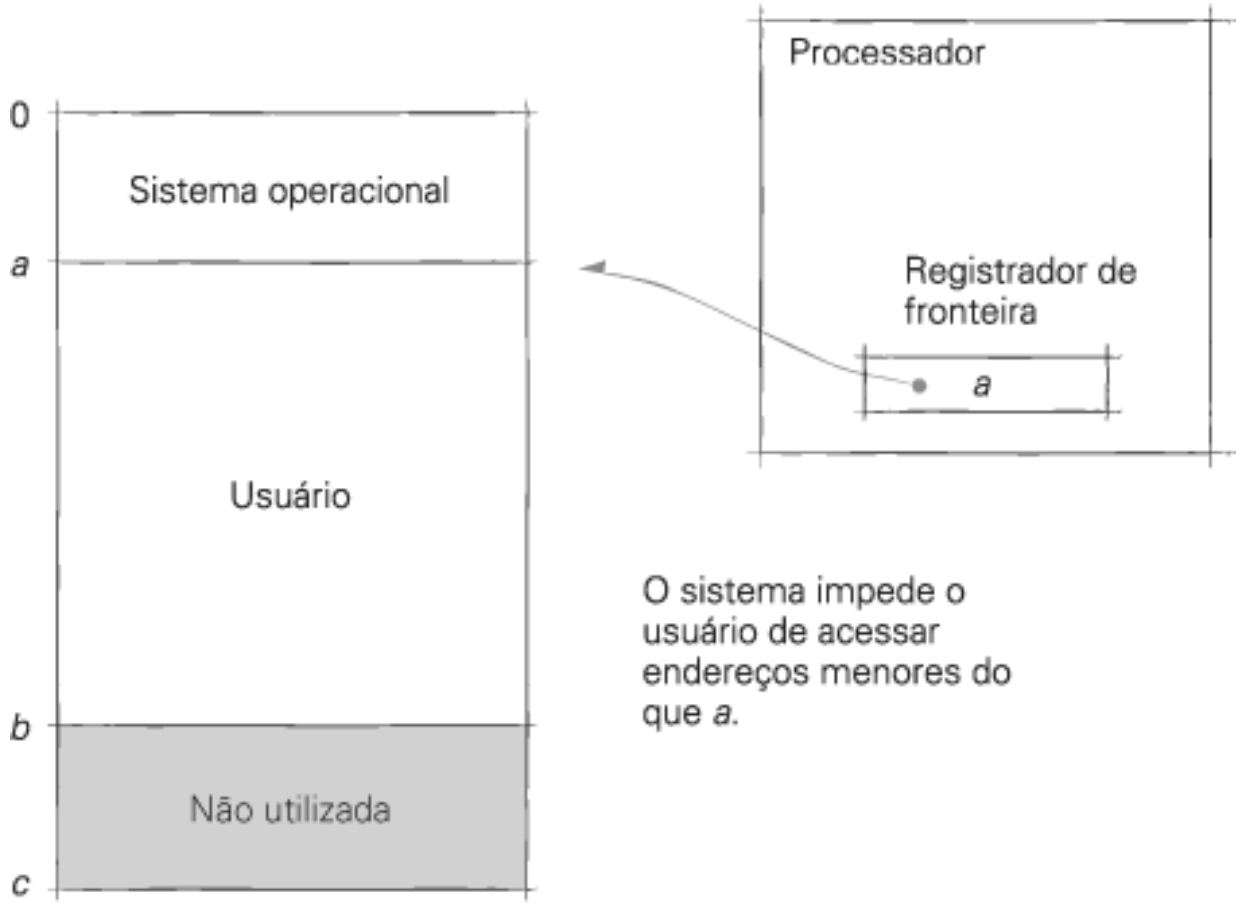
9.7.2 Proteção em um sistema monousuário

- **O sistema operacional não deve ser prejudicado pelos programas**
  - O sistema não conseguirá funcionar se o sistema operacional for sobregravado.
  - Registrador de fronteira
    - Contém o endereço em que o espaço de memória do programa inicia.
    - Todo acesso à memória, fora dessa fronteira, é negado.
    - Só pode ser definido por comandos privilegiados.
    - As aplicações podem acessar a memória do sistema operacional para executar procedimentos por meio de chamadas ao sistema, o que coloca o sistema no modo executivo.

PEARSON Prentice Hall

9.7.2 Proteção em um sistema monousuário

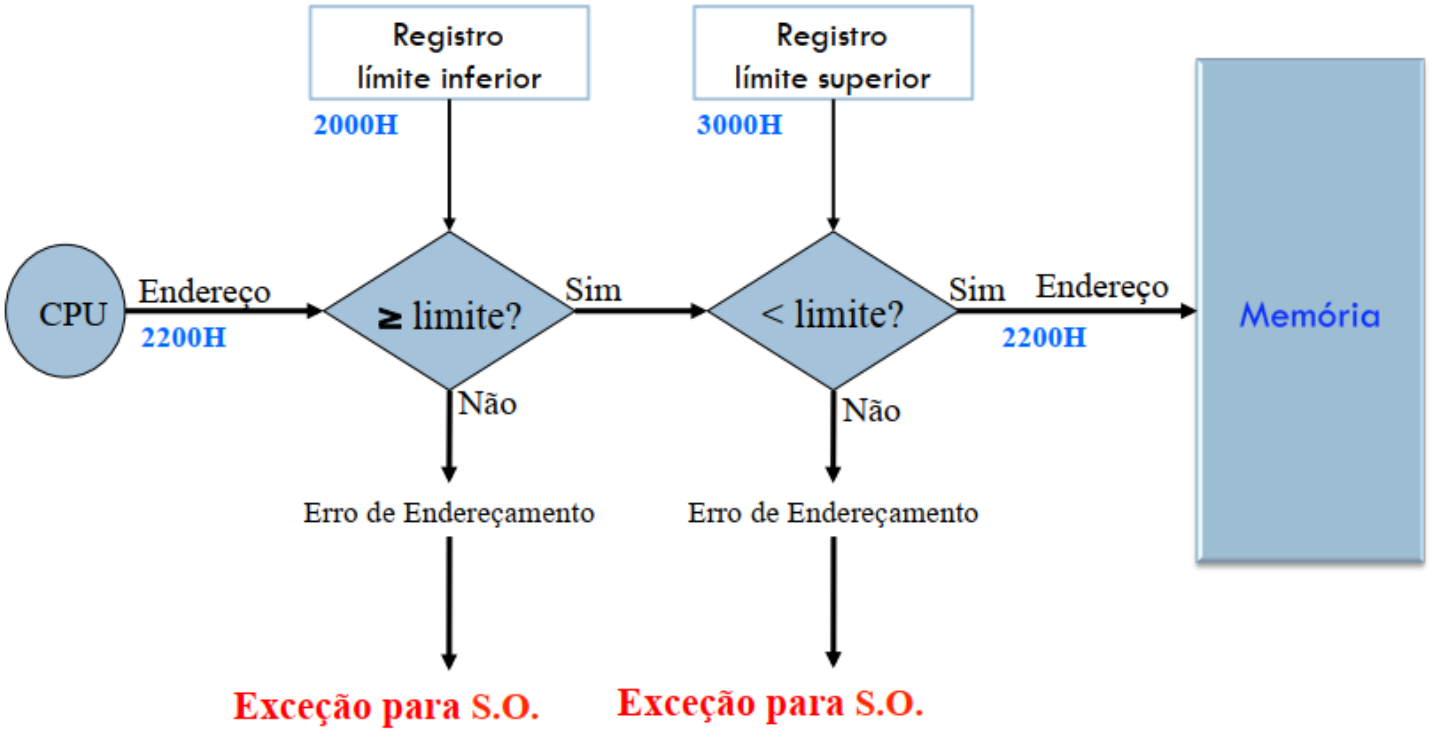
Figura 9.5 Proteção de memória com alocação de memória contígua em sistema monousuário.



9.7.2 Proteção em um sistema monousuário

Figura 9.5 Proteção de memória com alocação de memória contígua em sistema monousuário.

Hardware de Proteção



9.7.3 Processamento em lote de fluxo único

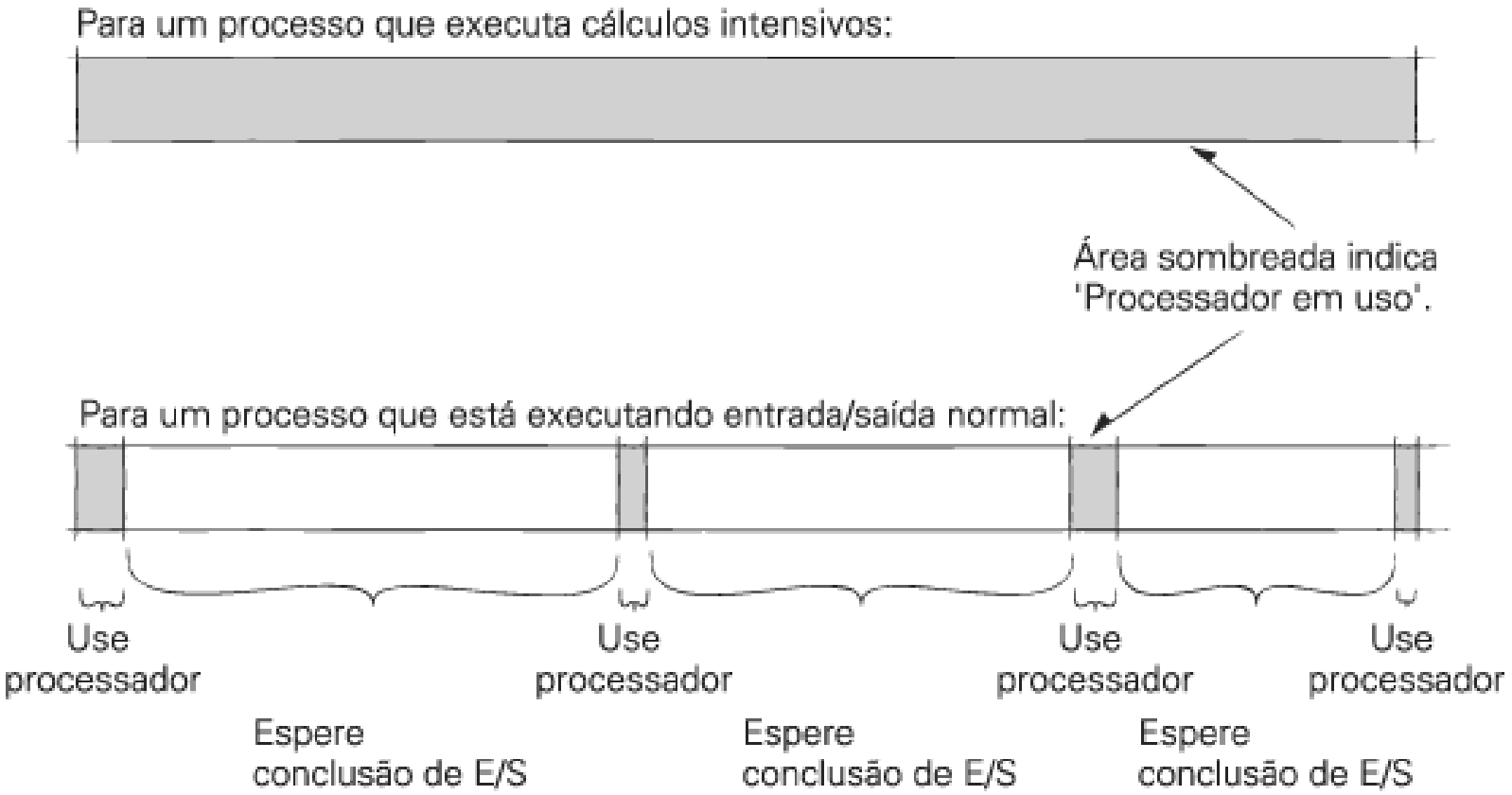
- **Os primeiros sistemas exigiam muito tempo de preparação.**
  - Tempo e recursos desperdiçados.
  - Maior eficiência com a automatização da preparação e da desmontagem.
- **Processamento em lote**
  - O processador de fluxo de jobs lê as linguagens de controle de jobs.
    - Definem cada job e como ele deve ser configurado.

9.8 Multiprogramação por partição fixa

- **As solicitações E/S podem reter o processador por longos períodos.**
  - A multiprogramação é uma das soluções.
    - O processo que não usa constantemente o processador deve cedê-lo a outros.
    - Exige que diversos processos permaneçam na memória simultaneamente.

9.8 Multiprogramação por partição fixa

Figura 9.6 Utilização do processador em um sistema monousuário. [Nota: Em muitos jobs monousuário, as esperas de E/S são muito mais longas em relação aos períodos de utilização do processador indicados neste diagrama.]



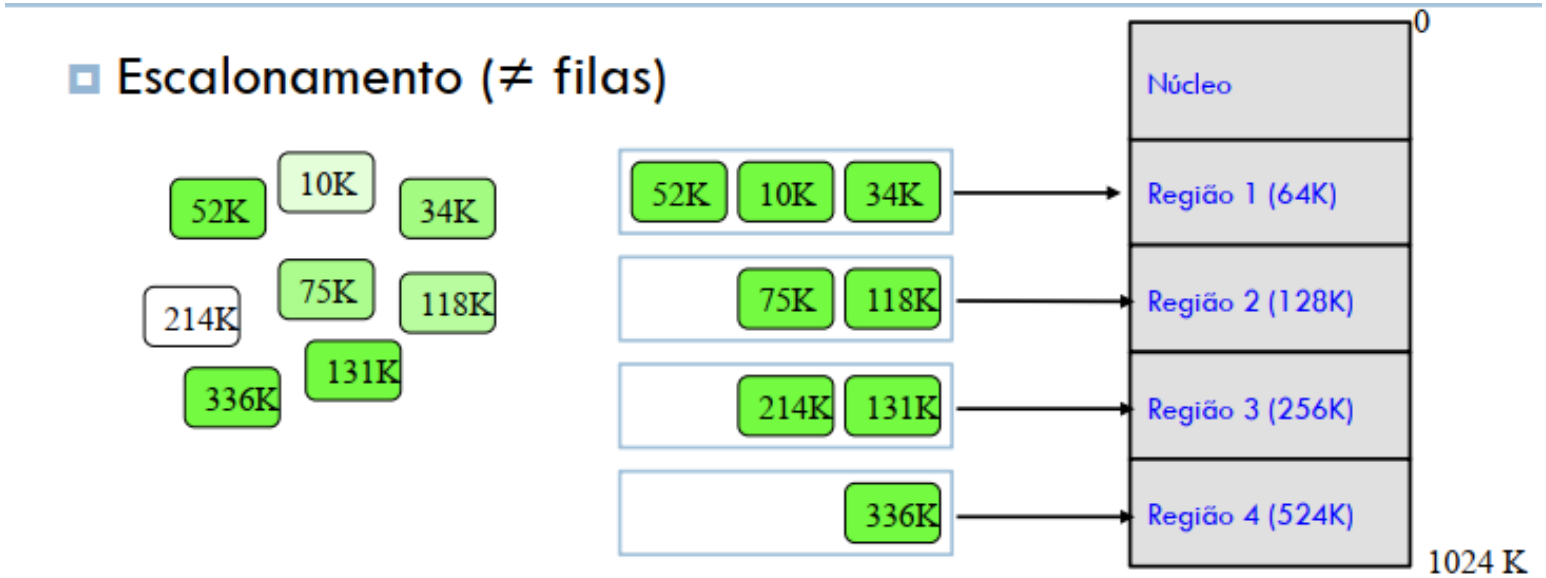
9.8 Multiprogramação por partição fixa

- **Multiprogramação por partição fixa**
  - Todo processo ativo recebe um bloco de tamanho único da memória.
  - O processador alterna rapidamente de um processo para o outro.
  - Vários registradores de fronteira oferecem proteção contra prejuízos.

PEARSON Prentice Hall

9.8 Multiprogramação por partição fixa

Figura 9.7 Multiprogramação por partição fixa com tradução e carregamento absolutos.



- Desvantagens:
  - Trabalho não balanceado: Determinadas partiçoes podem ficar sem tarefas, enquanto outras tarefas ficam esperando em fila.
- Vantagens:
  - Mais eficiente: Melhor utilização dos recursos de memória
  - Hardware **mais simples:** Só necessita relocação estática em tempo de compilação.

9.8 Multiprogramação por partição fixa

- **Inconvenientes das partições fixas:**
  - Antes, as implementações usavam endereços absolutos.
    - Se a partição solicitada estivesse cheia, o código não conseguia carregar.
    - Posteriormente, isso foi solucionado pelos compiladores de realocação.

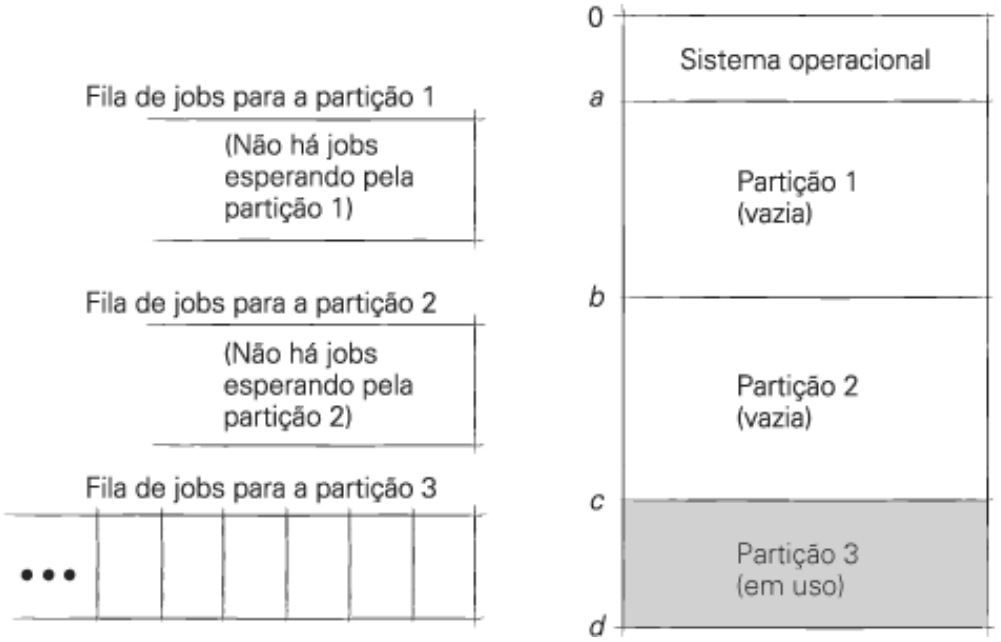
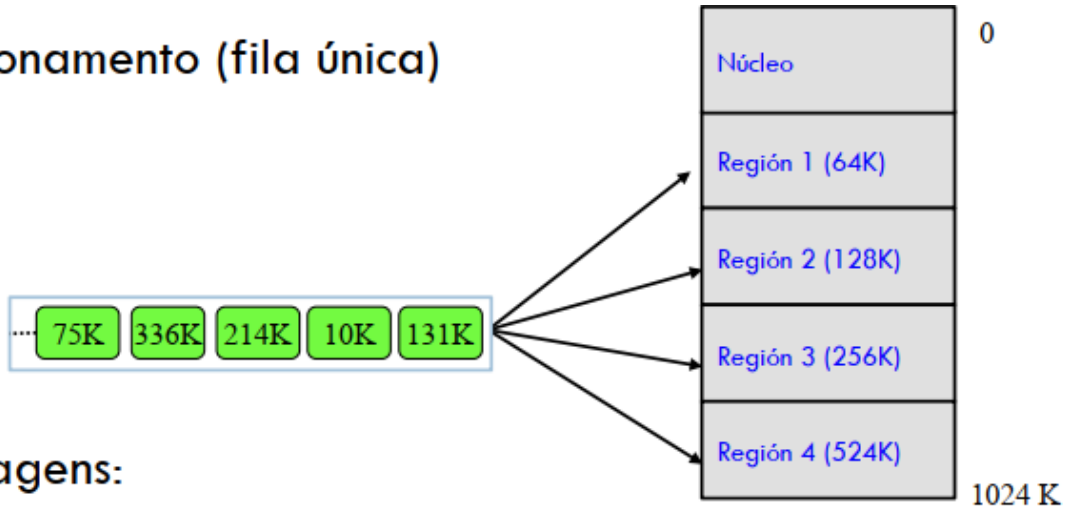


Figura 9.8 Desperdício de memória na multiprogramação por partição fixa com tradução e carregamento absolutos.

9.8 Multiprogramação por partição fixa

Figura 9.9 Multiprogramação por partição fixa com tradução e carregamento realocáveis.

- Escalonamento (fila única)



- Vantagens:
  - Maior grau de multi-programação: Permite executar um maior número de processos de forma simultânea
  - Melhor balanceamento de carga.
- Desvantagens:
  - Incrementa a quantidade de memória desaproveitada.
  - Novos Requerimentos: Relocação estática em tempo de execução e políticas de seleção de partição.

9.8 Multiprogramação por partição fixa

- **Proteção**

  Pode ser implementada pelos registradores de fronteira, denominados base e limite (também chamados baixo e alto).

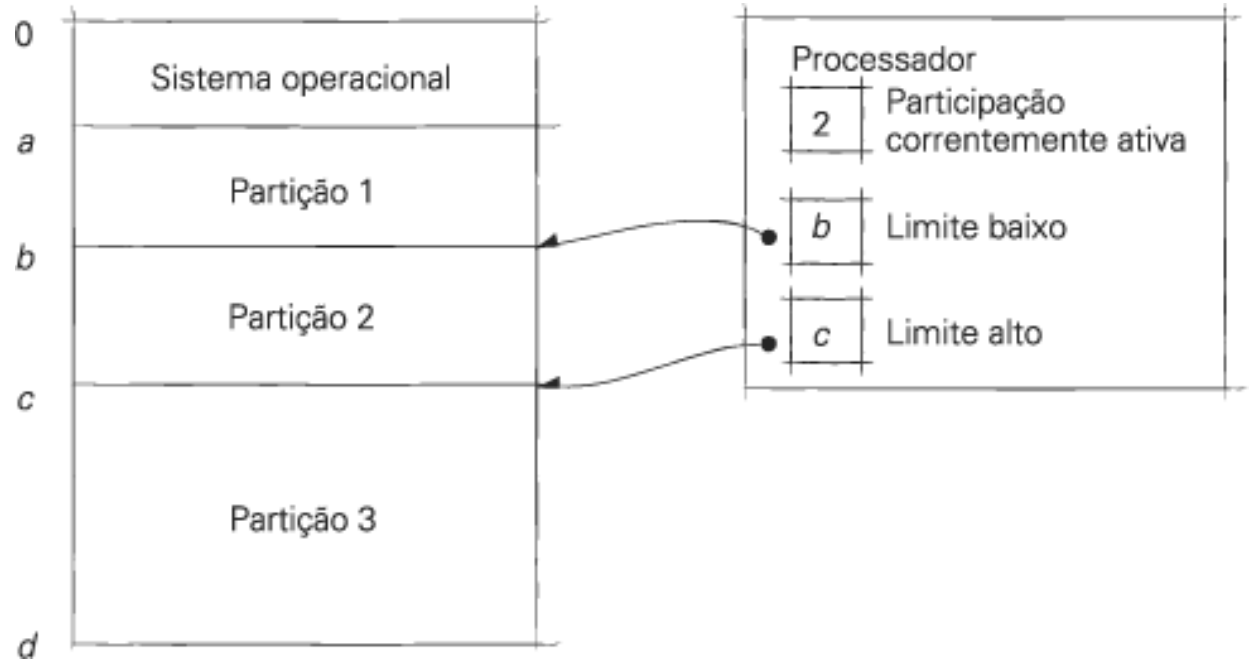


Figura 9.10 Proteção da memória em sistemas de multiprogramação de alocação contígua.

9.8 Multiprogramação por partição fixa

- **Inconvenientes das partições fixas (continuação)**
  - Fragmentação interna
    - O processo não ocupa a partição inteira, e isso desperdiça memória.
  - A sobrecarga é maior.
    - Isso é compensado por uma maior utilização dos recursos.

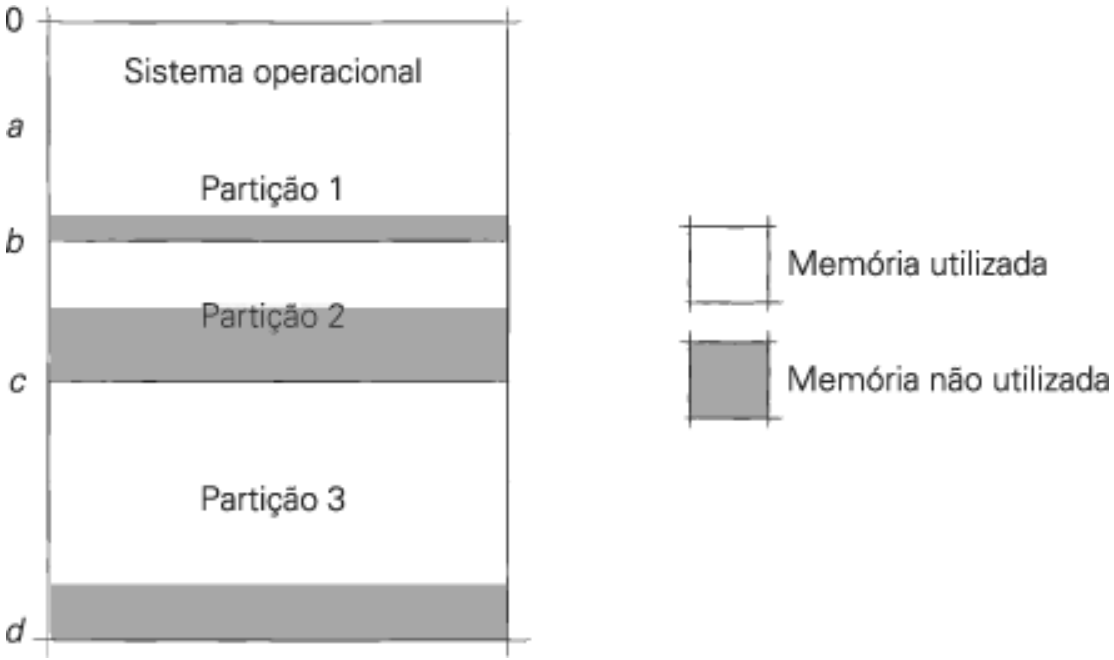
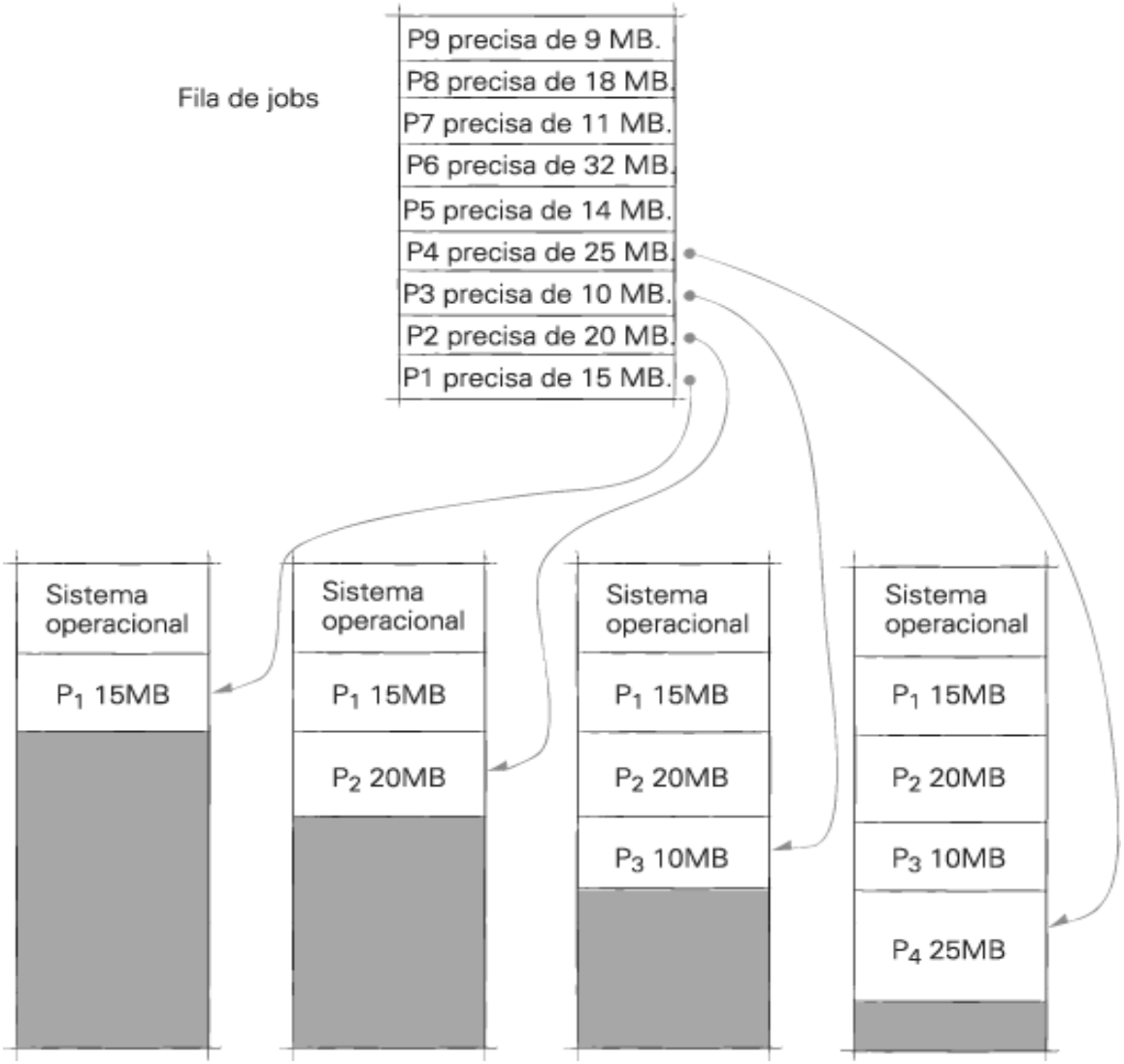


Figura 9.11 Fragmentação interna em um sistema de multiprogramação por partição fixa.

9.9 Multiprogramação por partição variável

Figura 9.12 Designações de partições iniciais na programação por partição variável.



9.9 Multiprogramação por partição variável

- **Para os projetistas de sistema, as partições fixas apresentam muitas restrições.**

  Existe a possibilidade de os processos ficarem muito grandes e não se encaixarem em nenhum lugar.

  Fragmentação interna.

  As partições variáveis foram desenvolvidas para substituí-las.

9.9.1 Características da partição variável

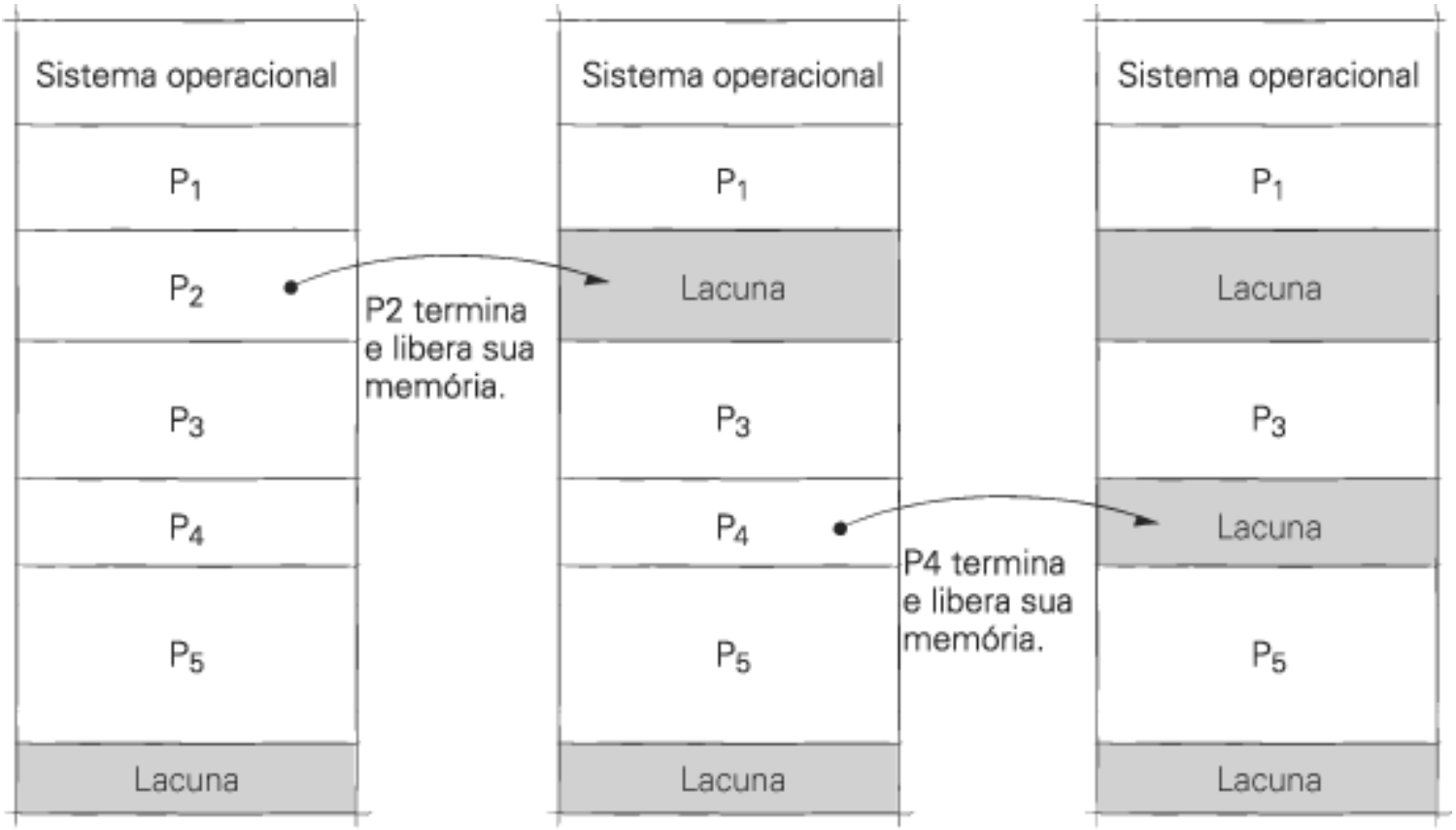
Os jobs são posicionados no lugar em que se encaixam.

- Inicialmente, nenhum espaço é desperdiçado.
- Não é possível haver fragmentação interna.
  - As partições têm exatamente o tamanho que precisam.
- Pode ocorrer fragmentação externa quando algum processo for removido.
  - As lacunas (espaços livres) deixadas para novos processos são muito pequenas.
  - Existe a possibilidade de não haver espaço livre suficiente para novos processos.

PEARSON Prentice Hall

9.9.1 Características da partição variável

Figura 9.13 "Lacunas" de memória em multiprogramação por partição variável.



9.9.1 Características da partição variável

- **Existem várias formas de combater a fragmentação externa.**

Coalescência

Reúne blocos livres adjacentes em um único bloco grande.

Em geral isso não é suficiente para obter uma quantidade significativa de memória.

Compactação

Às vezes é chamada coleta de lixo (que não deve ser confundida com a coleta de lixo nas linguagens orientadas a objeto).

Reorganização da memória em um único bloco contíguo de espaço livre e em um único bloco contíguo de espaço ocupado.

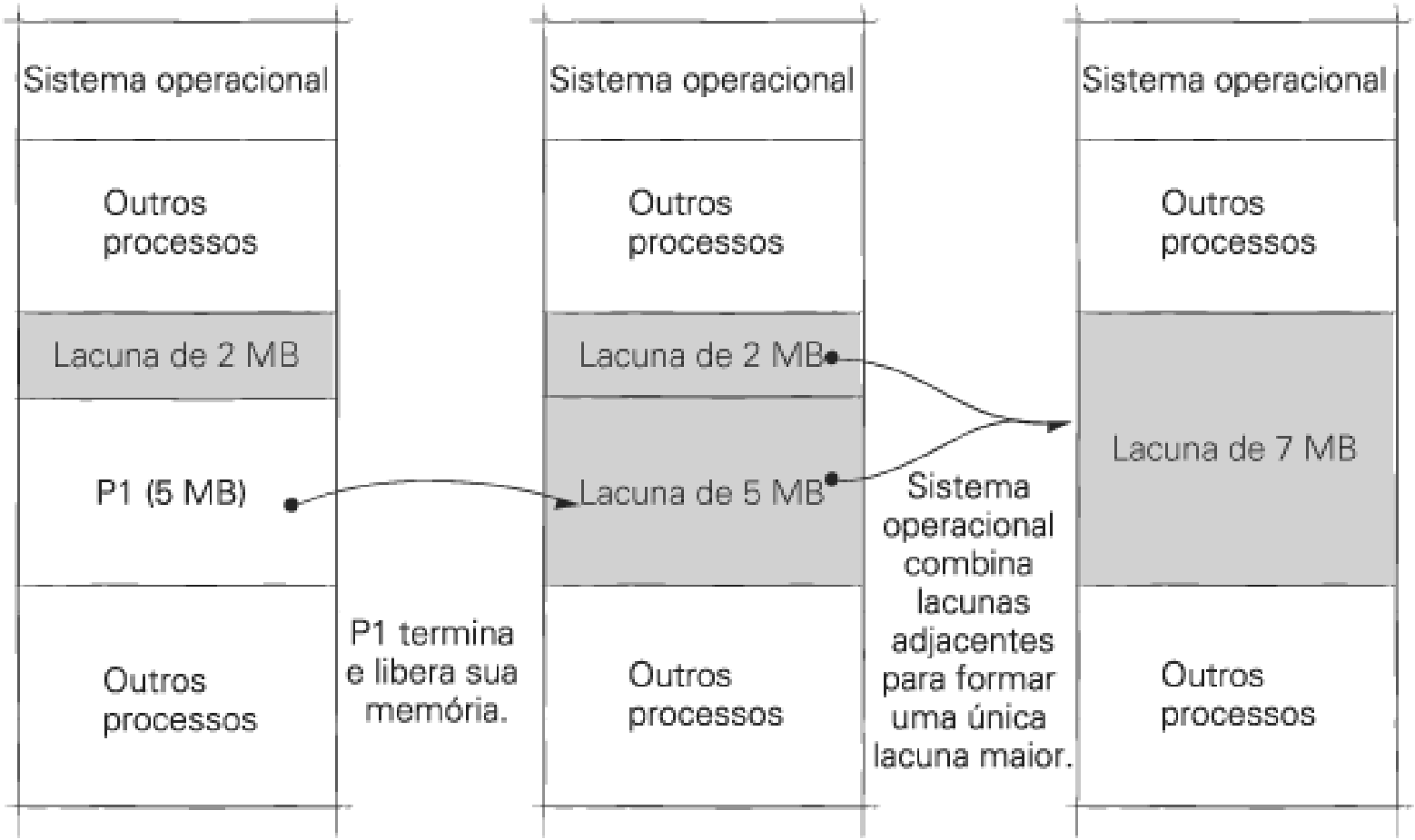
Possibilita que todos espaços livres fiquem disponíveis.

A sobrecarga é significativa.

PEARSON Prentice Hall

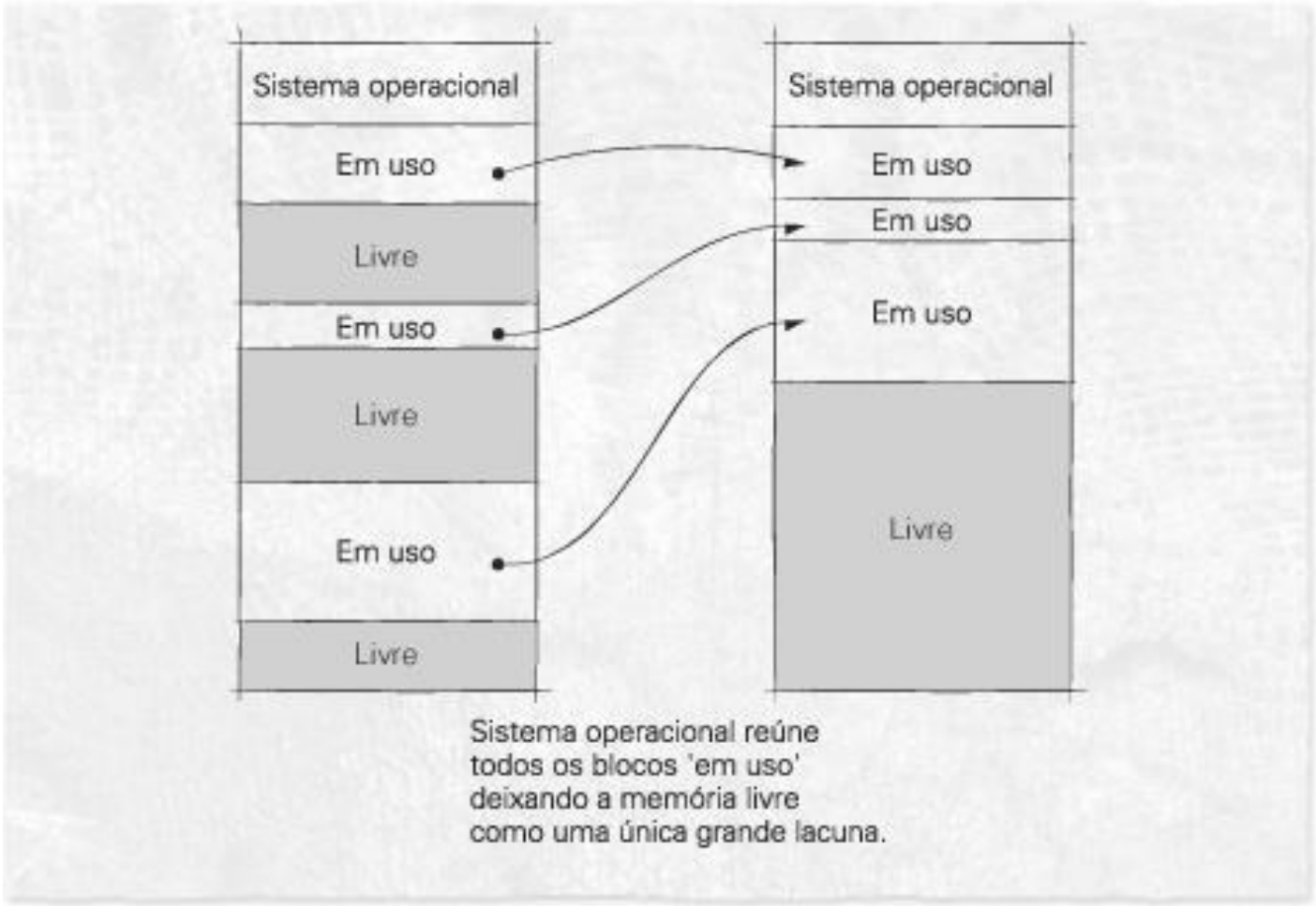
9.9.1 Características da partição variável

Figura 9.14 Coalescência de "lacunas" de memória em multiprogramação por partição variável.



9.9.1 Características da partição variável

Figura 9.15 Compactação de memória em multiprogramação por partição variável.



9.9.2 Estratégias de posicionamento de memória

- **Onde posicionar os processos que estão chegando**

Estratégia do primeiro que couber

O processo é posicionado na primeira lacuna de tamanho suficiente encontrada.

Sobrecarga baixa e elementar em tempo de execução.

Estratégia o que melhor couber

O processo é posicionado na lacuna que deixar o menor espaço não utilizado ao seu redor.

Maior sobrecarga em tempo de execução.

Estratégia o que pior couber

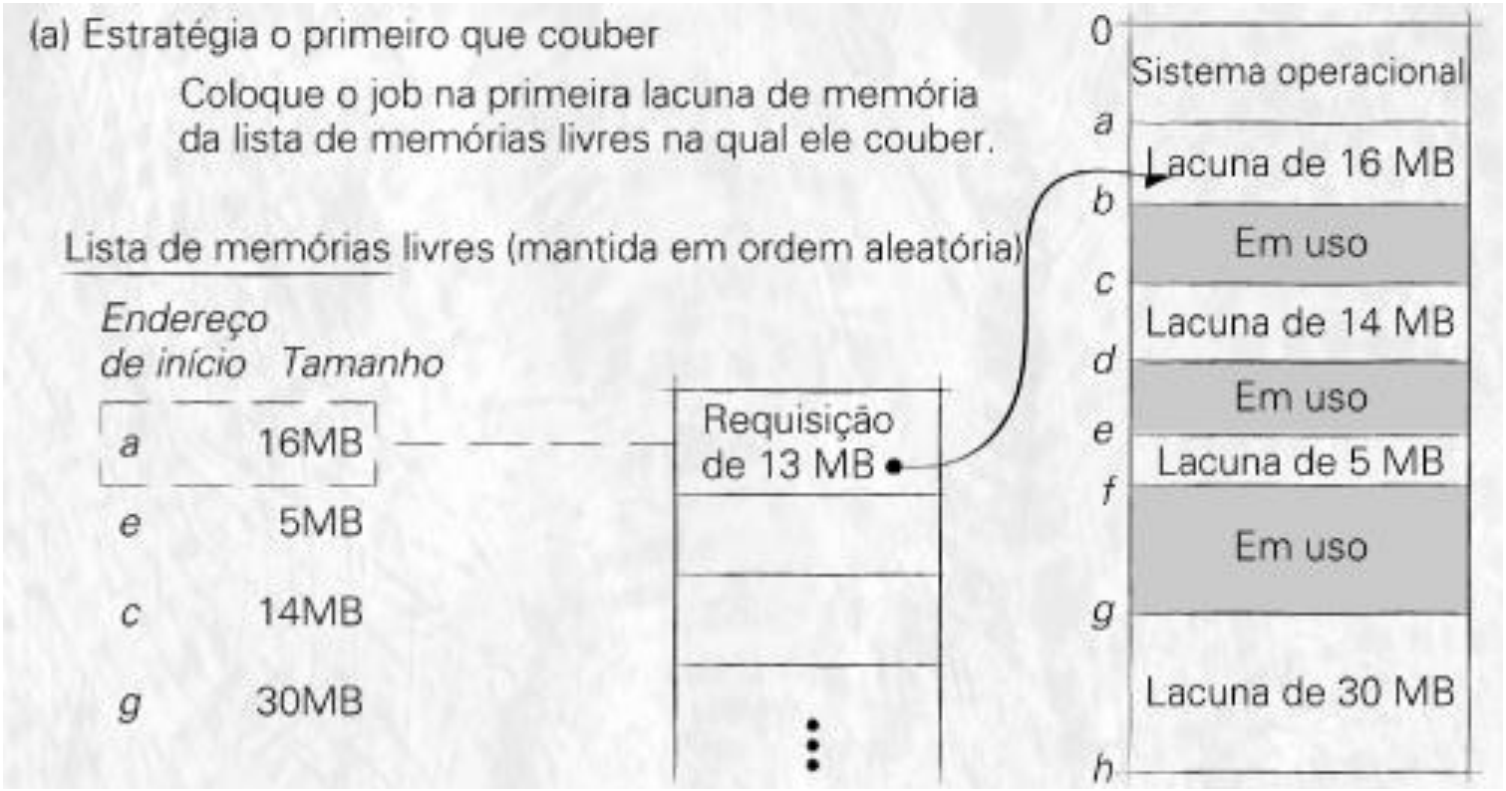
O processo é posicionado na lacuna que deixar o maior espaço não utilizado ao seu redor.

Nesses casos, é deixada uma outra grande lacuna, o que permite que outro processo caiba nessa lacuna.

PEARSON Prentice Hall

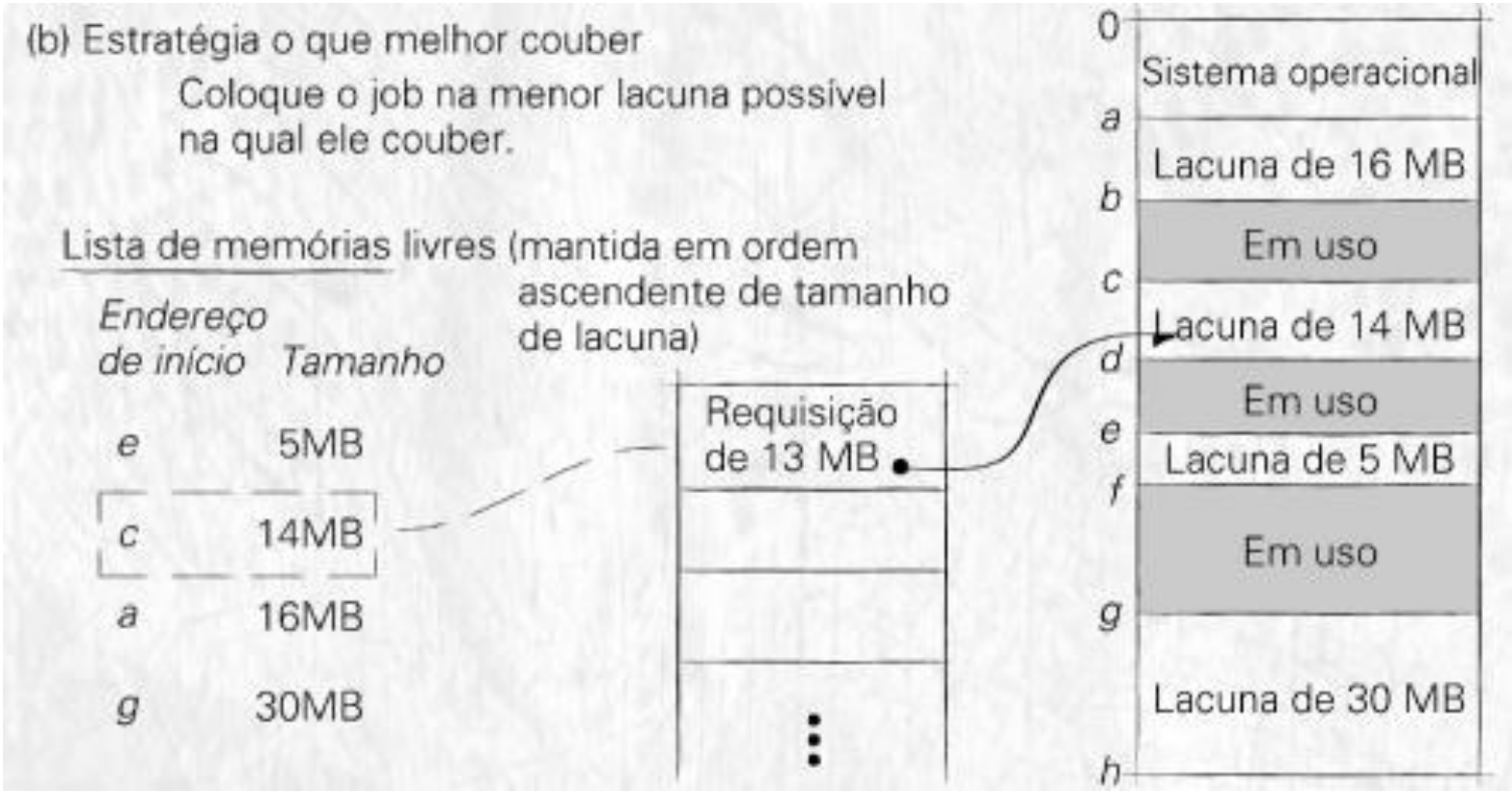
9.9.2 Estratégias de posicionamento de memória

Figura 9.16 Estratégias de posicionamento na memória o primeiro que couber, o que melhor couber, o que pior couber (Parte 1 de 3).



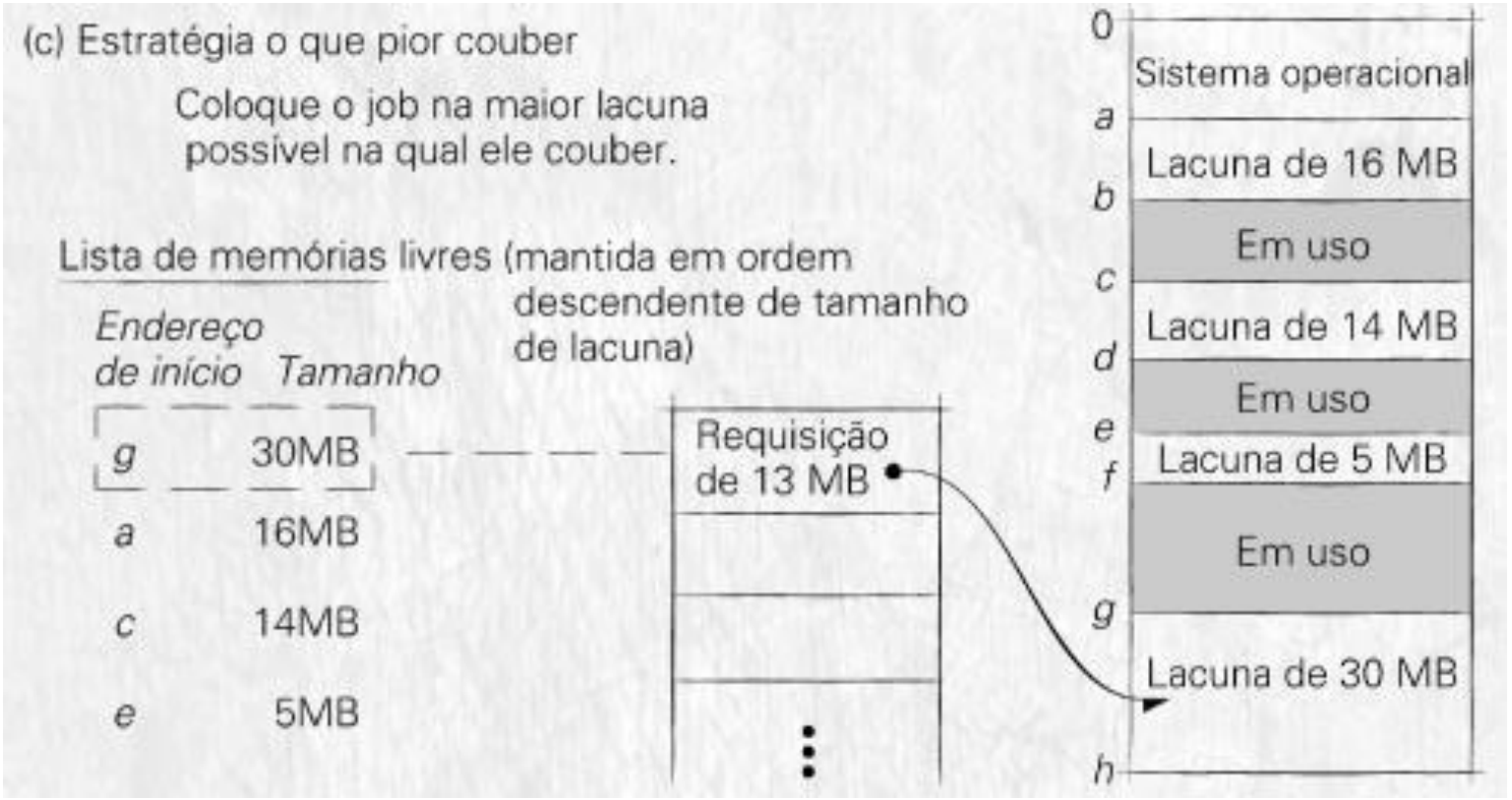
9.9.2 Estratégias de posicionamento de memória

Figura 9.16 Estratégias de posicionamento na memória o primeiro que couber, o que melhor couber, o que pior couber (Parte 2 de 3).



9.9.2 Estratégias de posicionamento de memória

Figura 9.16 Estratégias de posicionamento na memória o primeiro que couber, o que melhor couber, o que pior couber (Parte 3 de 3).



9.11 Estratégias para Escolha da Partição

Best-fit

- Escolhe-se a partição onde o processo deixa o menor espaço sem utilização;
- Escolhe o bloco cujo tamanho é o mais próximo do requisitado.

Worst-fit

- Escolhe-se a partição onde o processo deixa o maior espaço sem utilização;
- Escolhe o maior espaço livre na memória.

First-fit

- Busca por espaço livre
  - Varre a memória do início
  - Escolhe o primeiro bloco disponível que seja grande o suficiente

Next-fit

- Similar ao First-fit;
- Diferença está na busca, que ocorre a partir do endereço da última posição alocada.

9.10 Multiprogramação com troca de memória (swapping)

- **Não há necessidade de manter processos inativos na memória.**

Troca (Swapping)

Coloca apenas projetos em execução na memória principal.

Outros são temporariamente transferidos para o armazenamento secundário.

Isso maximiza a memória disponível.

Há uma sobrecarga significativa quando do chaveamento de processos.

Melhor ainda: mantém diversos processos na memória ao mesmo tempo.

Menos memória disponível.

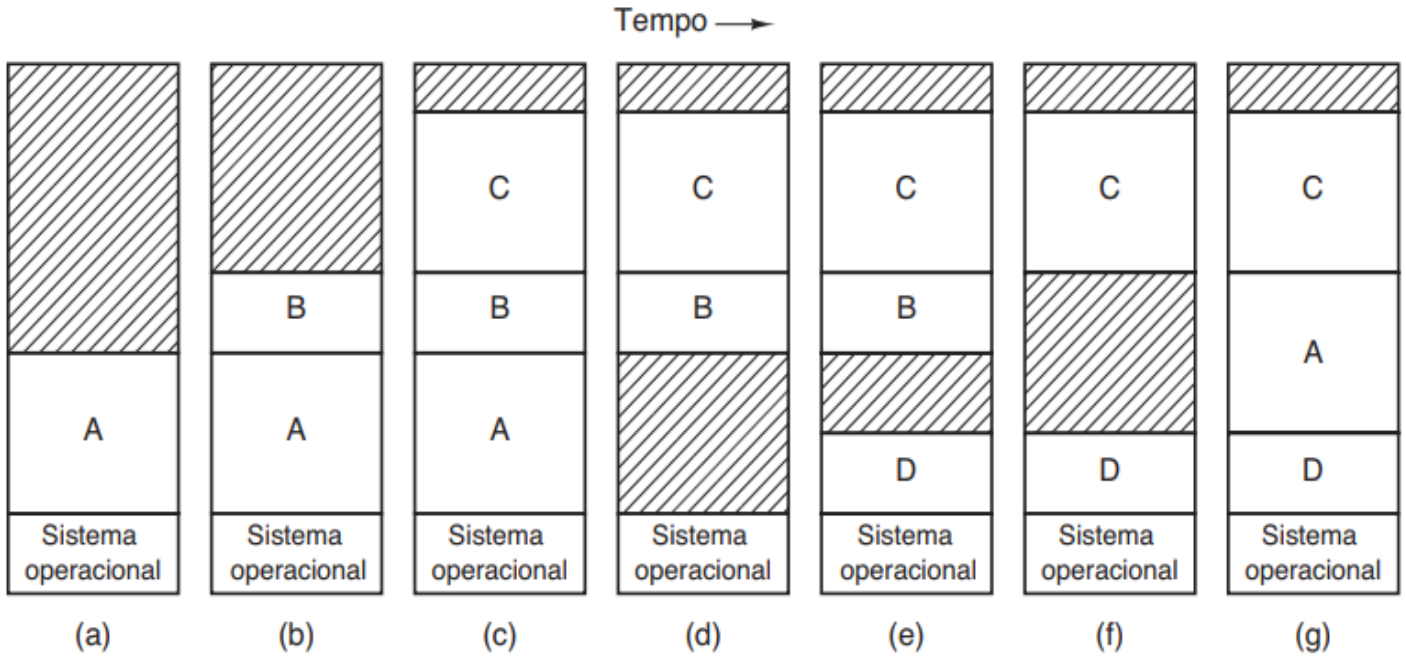
Tempos de resposta mais rápido.

Semelhante à paginação.

9.10 Multiprogramação com troca de memória (swapping)

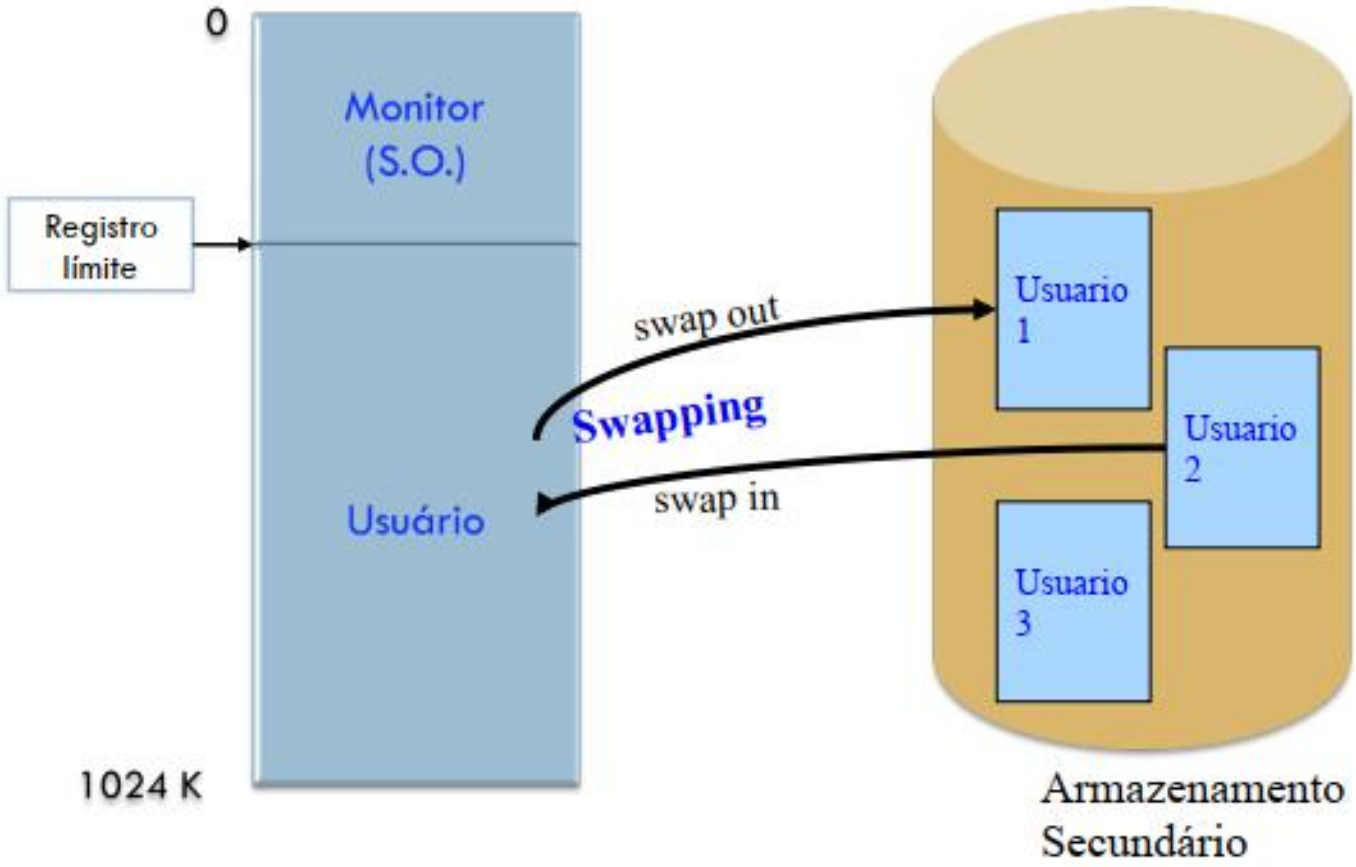
Figura 9.17 Multiprogramação em um sistema de troca de processos (swapping) no qual há somente um único processo por vez na memória principal.

FIGURA 3.4 Mudanças na alocação de memória à medida que processos entram nela e saem dela. As regiões sombreadas são regiões não utilizadas da memória.



9.10 Multiprogramação com troca de memória (swapping)

Figura 9.17 Multiprogramação em um sistema de troca de processos (swapping) no qual há somente um único processo por vez na memória principal.



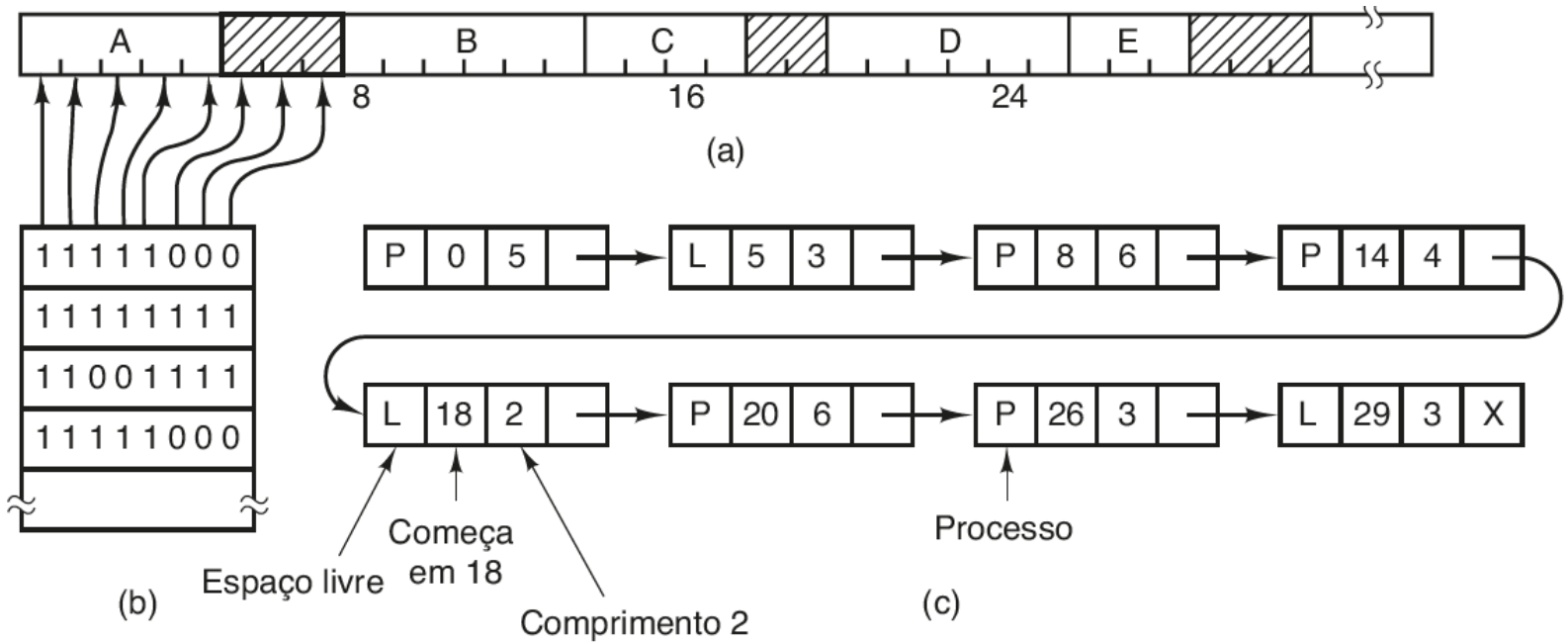
9.11 Estruturas para Gerenciar a memória

- **Usando Mapa de Bits (Bitmaps)**
  - Memória é dividida em unidades de alocação;
  - Unidade pode conter vários KB;
  - Cada unidade corresponde a um bit no bitmap:
    - 0 -> Livre
    - 1 -> ocupado

- **Lista Encadeada**
  - Manter uma lista ligada de segmentos de memória livres e alocados.

9.11 Gerenciamento de memória com mapas de bits

Figura 9.18 (a) Uma parte da memória com cinco processos e três espaços. As marcas indicam as unidades de alocação de memória. As regiões sombreadas (0 no mapa de bits) estão livres. (b) Mapa de bits correspondente. (c) A mesma informação como lista.



L = Espaço livre ou H = Hole

Exercícios

01. Considere um espaço de endereçamento lógico de oito páginas de 1024 palavras de 2 bytes cada, mapeado em uma memória física de 32 quadros.

a) Quantos bits tem o endereço lógico?

b) Quantos bits tem o endereço físico?

Para endereçar uma informação dentro de uma página de 1k

Precisamos de 10 bits. (2^{10} = 1024 posições)

Para endereçar as 8 páginas

Precisamos de 3 bits (2^3 = 8 posições)

Para endereçar os 32 frames

Precisamos de 5 bits (2^5 = 32 posições)

a) 13

b) 15

Exercícios

02. Considere um sistema cuja gerência de memória é feita através de partições variáveis. Nesse momento, existem as seguintes lacunas:

10k, 4k, 20k, 18k, 7k, 9k, 12k e 13k, nessa ordem.

Quais espaços serão ocupados pelas solicitações: 5k, 10k e 6k, nessa ordem, se:

a) First-fit for utilizado?

b) Next-fit for utilizado?

c) Best-fit for utilizado?

d) Worst-fit for utilizado?

PEARSON Prentice Hall

Exercícios

Lacunas: 10k, 4k, 20k, 18k, 7k, 9k, 12k e 13k

Solicitações: 5k, 10k e 6k

First-fit:

* **5k**, 4k, 20k, 18k, 7k, 9k, 12k e 13k
* 5k, 4k, **10k**, 18k, 7k, 9k, 12k e 13k
* 5k, 4k, **4k**, 18k, 7k, 9k, 12k e 13k

Next-fit:

* **5k**, 4k, 20k, 18k, 7k, 9k, 12k e 13k
* 5k, 4k, **10k**, 18k, 7k, 9k, 12k e 13k
* 5k, 4k, 10k, **12k**, 7k, 9k, 12k e 13k

Best-fit:

*10k, 4k, 20k, 18k, **2k**, 9k, 12k e 13k
***(0k)**, 4k, 20k, 18k, 2k, 9k, 12k e 13k
*(0k), 4k, 20k, 18k, 2k, **3k**, 12k e 13k

Worst-fit:

* 10k, 4k, **15k**, 18k, 7k, 9k, 12k e 13k
* 10k, 4k, 15k, **8k**, 7k, 9k, 12k e 13k
* 10k, 4k, **9k**, 8k, 7k, 9k, 12k e 13k

PEARSON Prentice Hall