Sistemas Operacionais I

Gerência de Memória: Memória Virtual

Prof. Alexandre Duarte : http://alexandrend.com
Centro de Informática | Universidade Federal da Paraíba

Estes slides são baseados no material que acompanha o livro Operating Systems Concepts de Silberschatz, Galvin and Gagne

Objetivos

- Descrever os benefícios de um sistema de memória virtual
- Explicar os conceitos de paginação sob demanda, algoritmos de substituição de páginas e alocação de quadros

Motivação

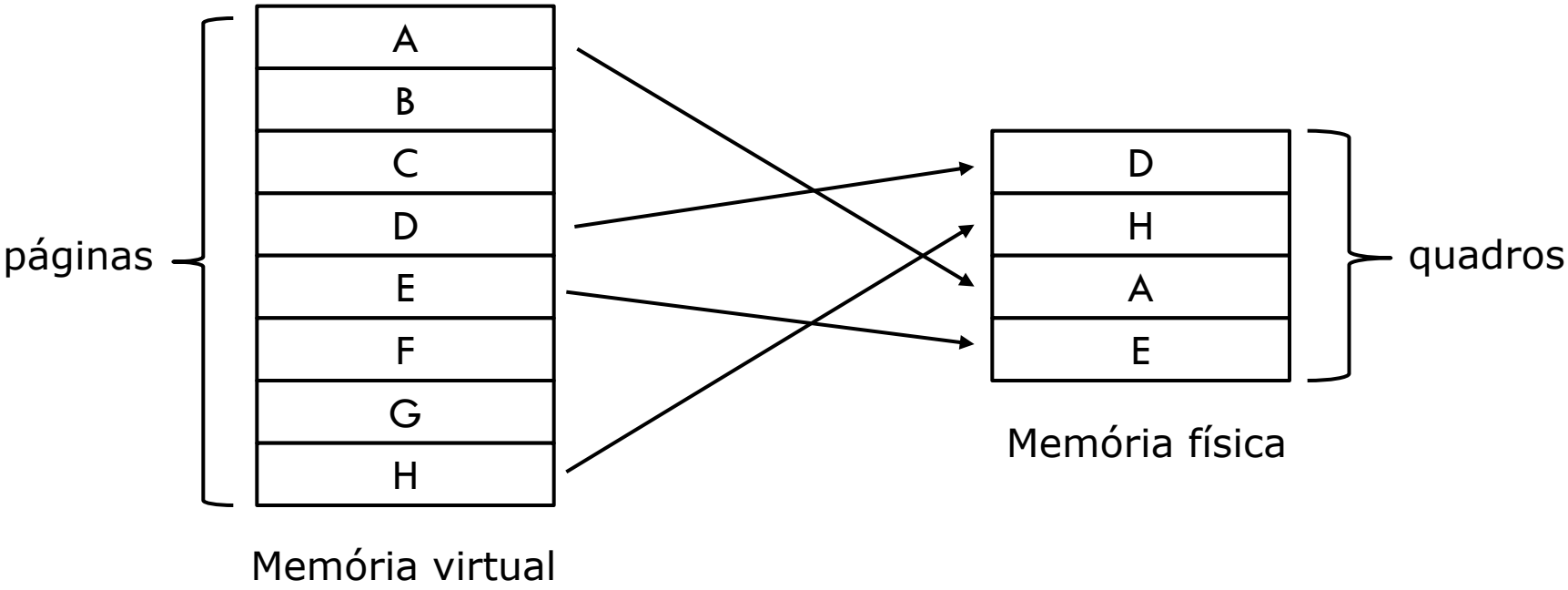
- Por que precisa da alocação contínua, paginação e segmentação?
  - Porque as instruções tem que estar na memória para serem executadas
- Problema
  - Limita o tamanho de um programa à memória

Motivação

- Porém, na prática o programa completo não é necessário
  - Código para manipular condições de erro não usuais: quase nunca é executado
  - Alocação de memória para array, listas e tabelas mais do que necessário
  - Não ser necessário todo o programa o tempo todo

Motivação

- Vantagens de executar um programa que está parcialmente na memória
  - Programa não está mais limitado ao tamanho da memória física: Memória Virtual



Motivação

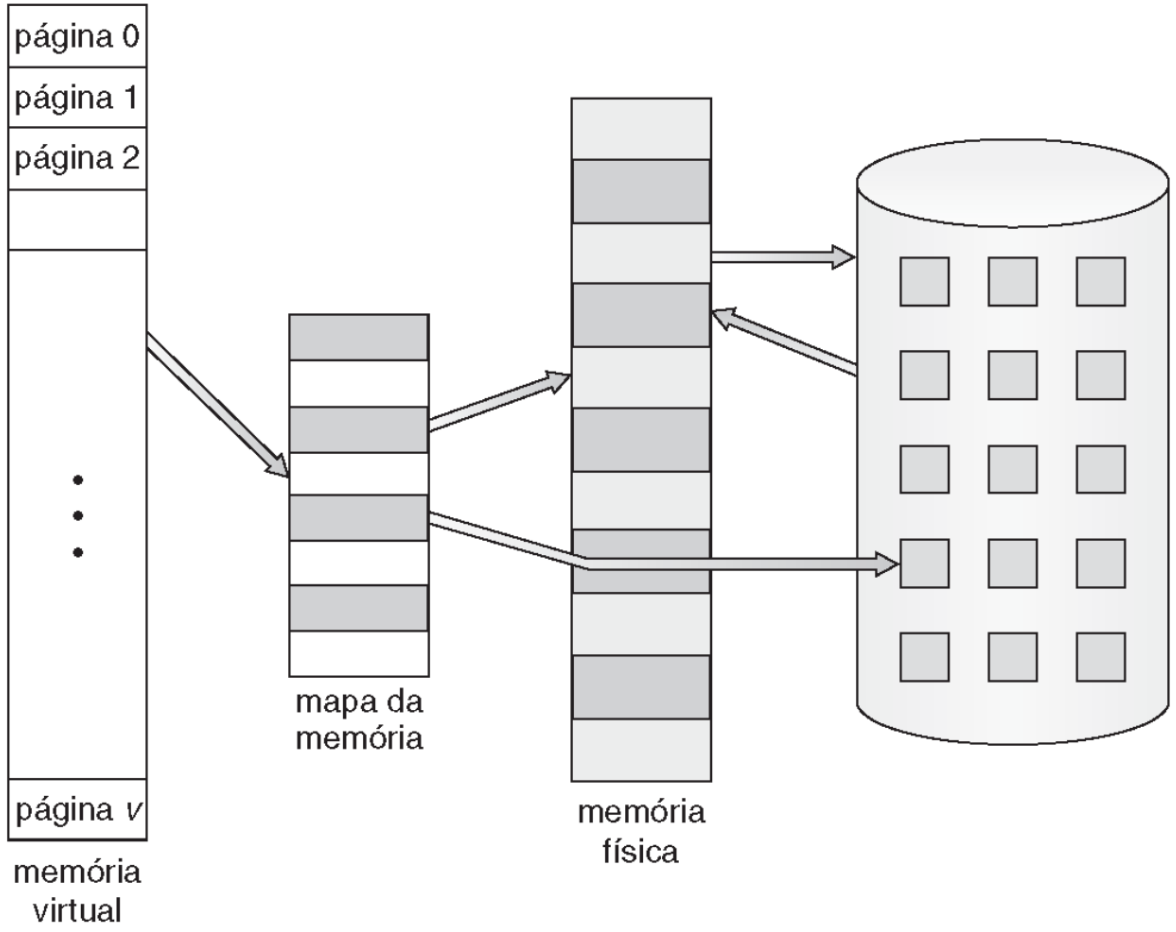
- Vantagens de executar um programa que está parcialmente na memória
  - Menor espaço de memória ocupado ➡ mais programas na memória ao mesmo tempo ➡ maior utilização de CPU (throughput) ➡ sem aumentar tempo de resposta
  - Menos E/S entre memória e disco

Contextualização

- **Memória virtual:** separação da memória lógica do usuário da memória física.
  - Apenas uma parte do programa precisa estar na memória para execução
  - Portanto, o espaço de endereçamento lógico pode ser muito maior que o espaço de endereçamento físico
  - Permite que os espaços de endereçamento sejam compartilhados por vários processos
  - Melhora desempenho na criação de processos (Copy-On-Write)
  - Programador “não precisa se preocupar” com quanto vai gastar de memória

- Memória virtual pode ser implementada de duas formas:
  - Paginação sob demanda
  - Segmentação sob demanda

Memória virtual maior que a memória física



Paginação sob demanda

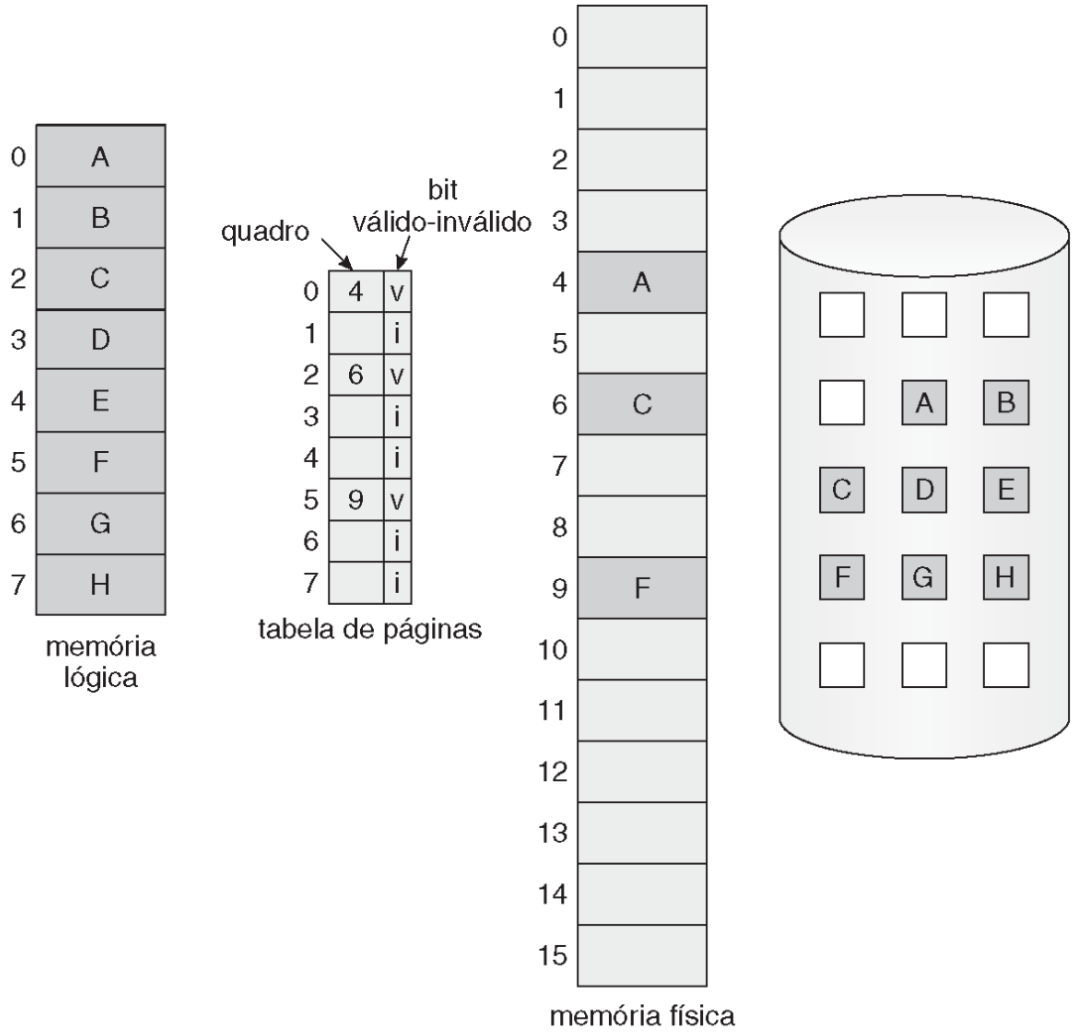
- Traz uma página para a memória apenas quando ela é necessária
  - Menos operações de E/S
  - Menor utilização de memória
  - Mais usuários/processos
  - Resposta mais rápida

- Página é necessária → referenciada
  - Referência inválida → aborta execução
  - Fora da memória → página é carregada

Bit válido/inválido

- Associa-se um bit de validade a cada entrada da tabela de paginação
- (**v** ⟶ na memória, **i** ⟶ fora da memória)
- Inicialmente setado para i para todas as entradas
- Durante a tradução de um endereço, se o bit vale **i** ⟶ falta de página!

Tabela de paginação com algumas páginas fora da memória principal



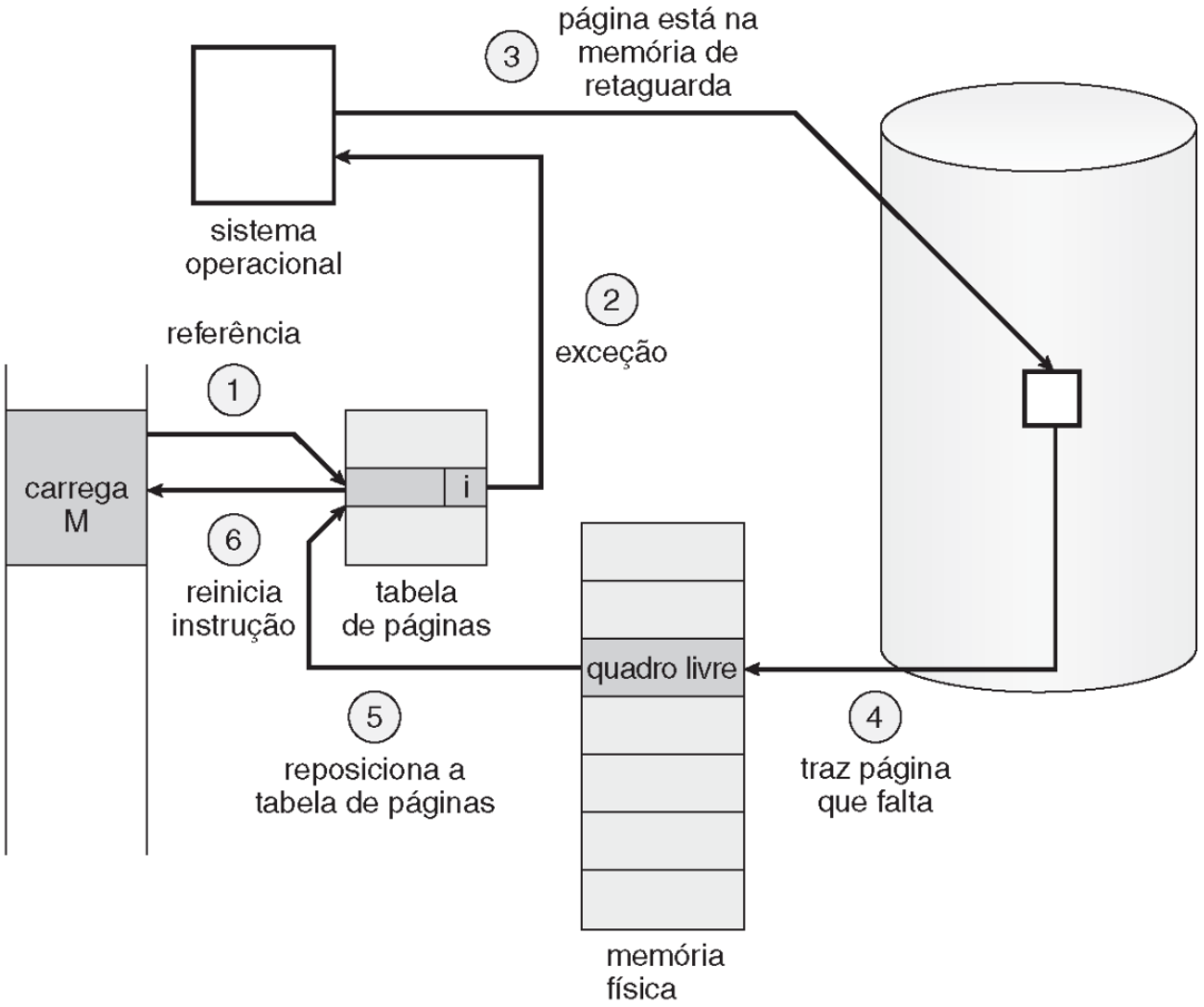
Falta de página

- A primeira referência a uma página fora da memória causa uma exceção:

  falta de página

1. O Sistema Operacional decide se houve uma:
   - Referência inválida → aborta o processo
   - Apenas uma referência a uma página fora da memória
2. Obtém um quadro livre
3. Carrega a página no quadro
4. Reconfigura as tabelas
5. Seta o bit de validade para **v**
6. Re-executa a instrução que gerou a falta de página

Sequência de passos para tratar uma falta de página



Desempenho da paginação sob demanda

- Taxa de falta de páginas $0 \leq p \leq 1.0$
  - se $p = 0$ não há falta de páginas
  - se $p = 1$, toda referência causa uma falta
- Tempo efetivo de acesso (EAT)

  EAT = (1 – *p*) x tempo de acesso a memória
  + *p* (**tempo para tratar falta de página** = tempo de swap out
  + tempo de carga da página
  + tempo para reiniciar a execução)

Exemplo de paginação sob demanda

- Tempo de acesso à memória = 200 nanossegundos

- Tempo médio para tratamento de uma falta de página = 8 milissegundos
- EAT = (1 – p) x 200 + p (8 milissegundos)

  = (1 – p) x 200 + p x 8,000,000

  = 200 – 200p + 800,000,000p

  = 200 + p x 7,999,800

- Se um acesso a cada 1.000 (0,1% = 0,001) causa uma falta de página então EAT = 8.2 microssegundos.

  40 vezes mais lento!!

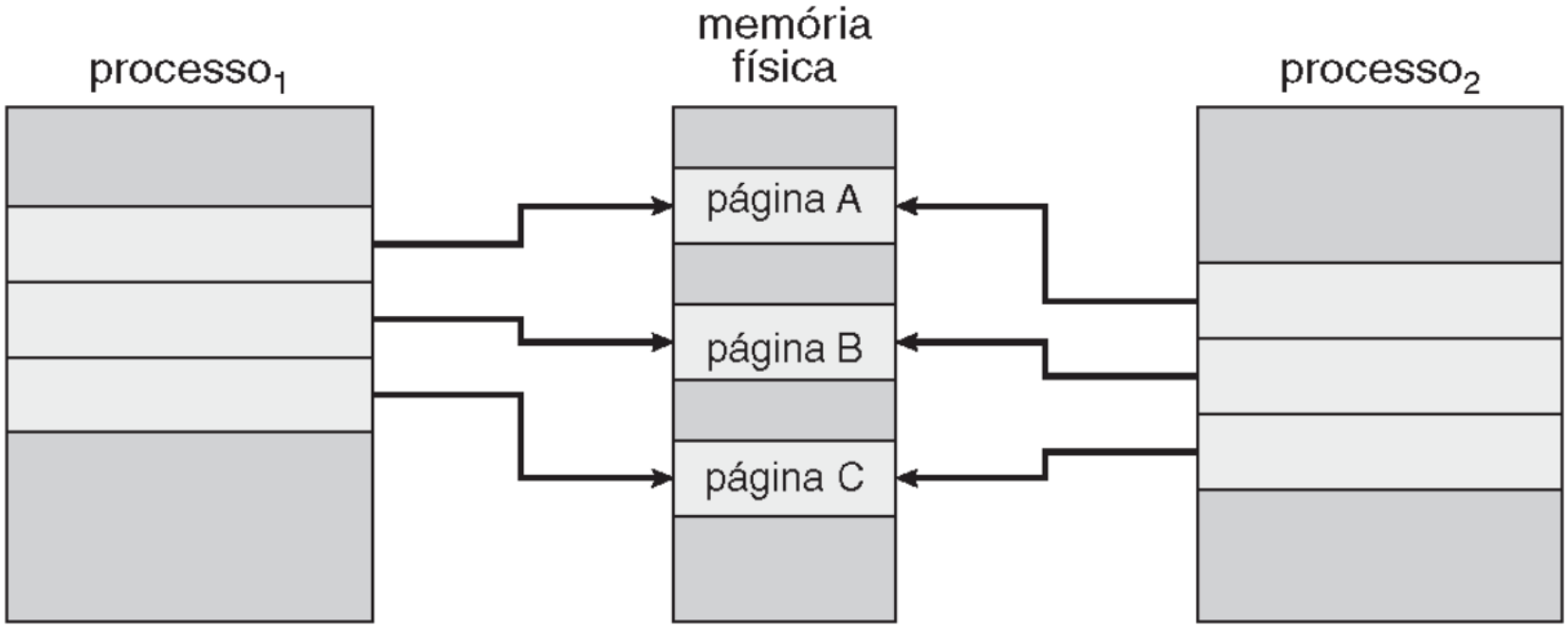
Criação de Processos

- O uso de memória virtual possibilita outros benefícios durante a criação de um processo:
  - Copy-on-Write
  - Arquivos mapeados em memória (mais tarde)

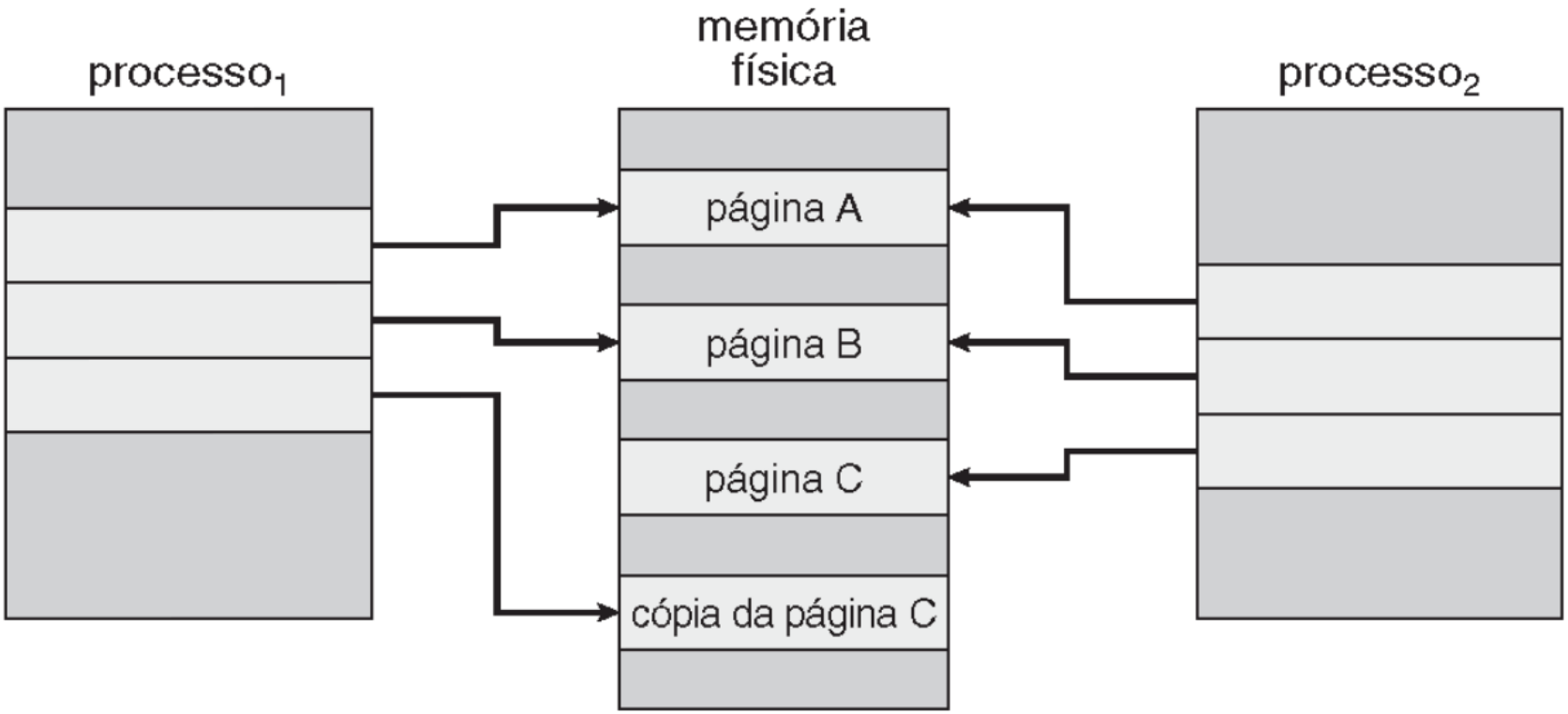
Copy-on-Write

- Copy-on-Write (COW) permite que processos pai e filho compartilhem inicialmente as mesmas páginas de memória
- Uma página só é copiada quando algum dos processos realiza uma operação de escrita
- COW permite uma criação mais eficiente de processo pois apenas páginas modificadas são copiadas

Antes do processo 1 modificar a página C



Após o processo 1 modificar a página C



O que acontece quando não há quadro livre?

- Substituição de página – encontra alguma página na memória mas que não esteja realmente em uso e a salva no disco (swap out)
  - algoritmo!
  - desempenho – queremos um algoritmo que resulte no menor número possível de faltas de página
  - Algumas páginas podem ser trazidas para a memória várias vezes

Substituição de página

- Impede que haja uma super-alocação de memória ao modificar a rotina de tratamento de falta de página para incluir a substituição
- Usa um bit de **modificação (dirty)** para reduzir o tempo de transferência – apenas páginas modificadas precisam ser gravadas no disco
- Substituição de páginas completa a separação entre memória lógica e memória física – uma memória virtual muito grande pode conviver com uma memória física bem menor

Processo básico de substituição

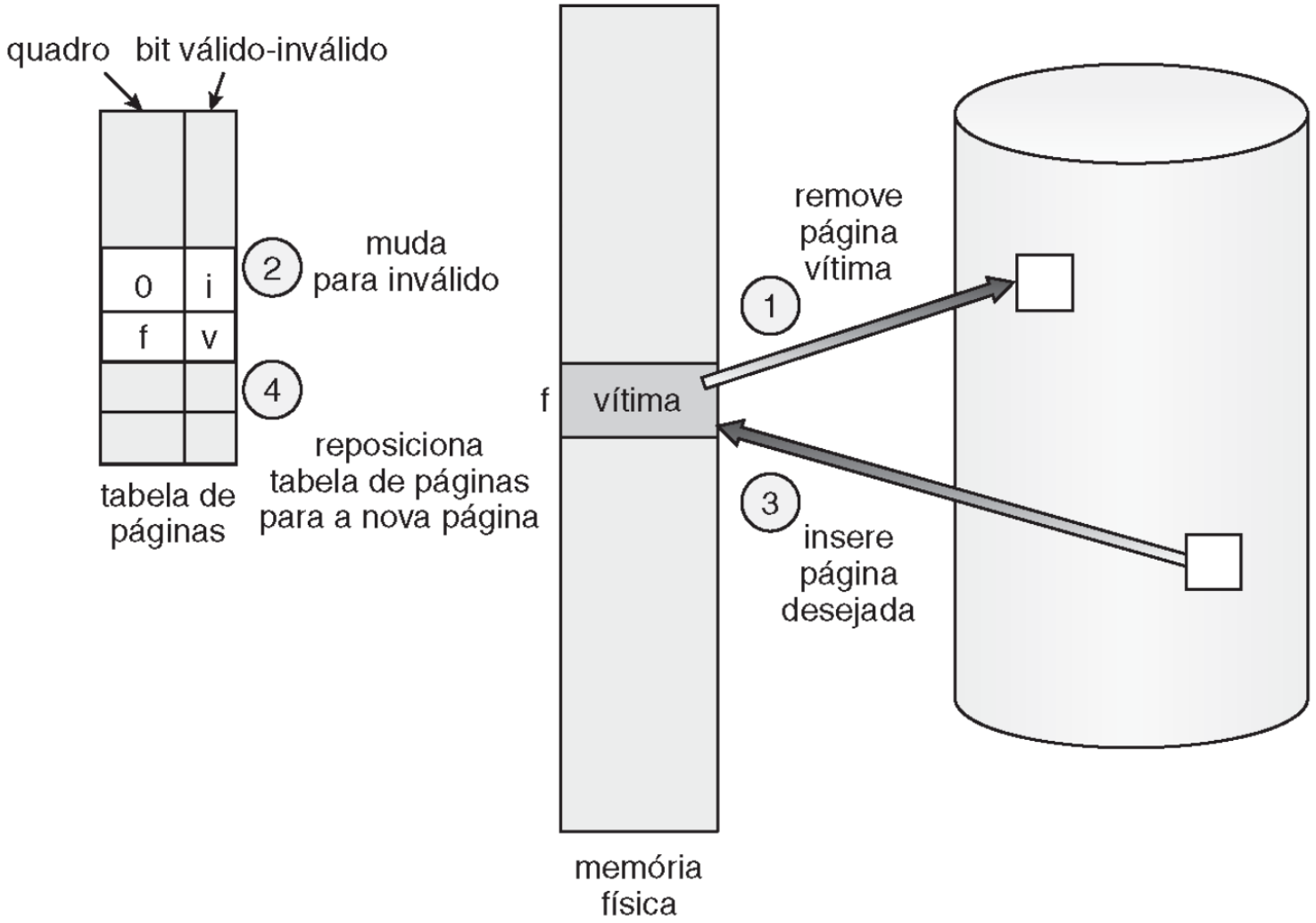
1. Localize a página desejada no disco

2. Localize um quadro livre:
   - Se há um quadro livre, utilize-o
   - Se não há tal quadro, utilizar um algoritmo de substituição para escolher um quadro vítima

3. Carregue a página desejada no quadro (recentemente) livre; atualize as tabelas de páginas e de quadros

4. Re-execute a instrução que causou a falta de página

Substituição de página



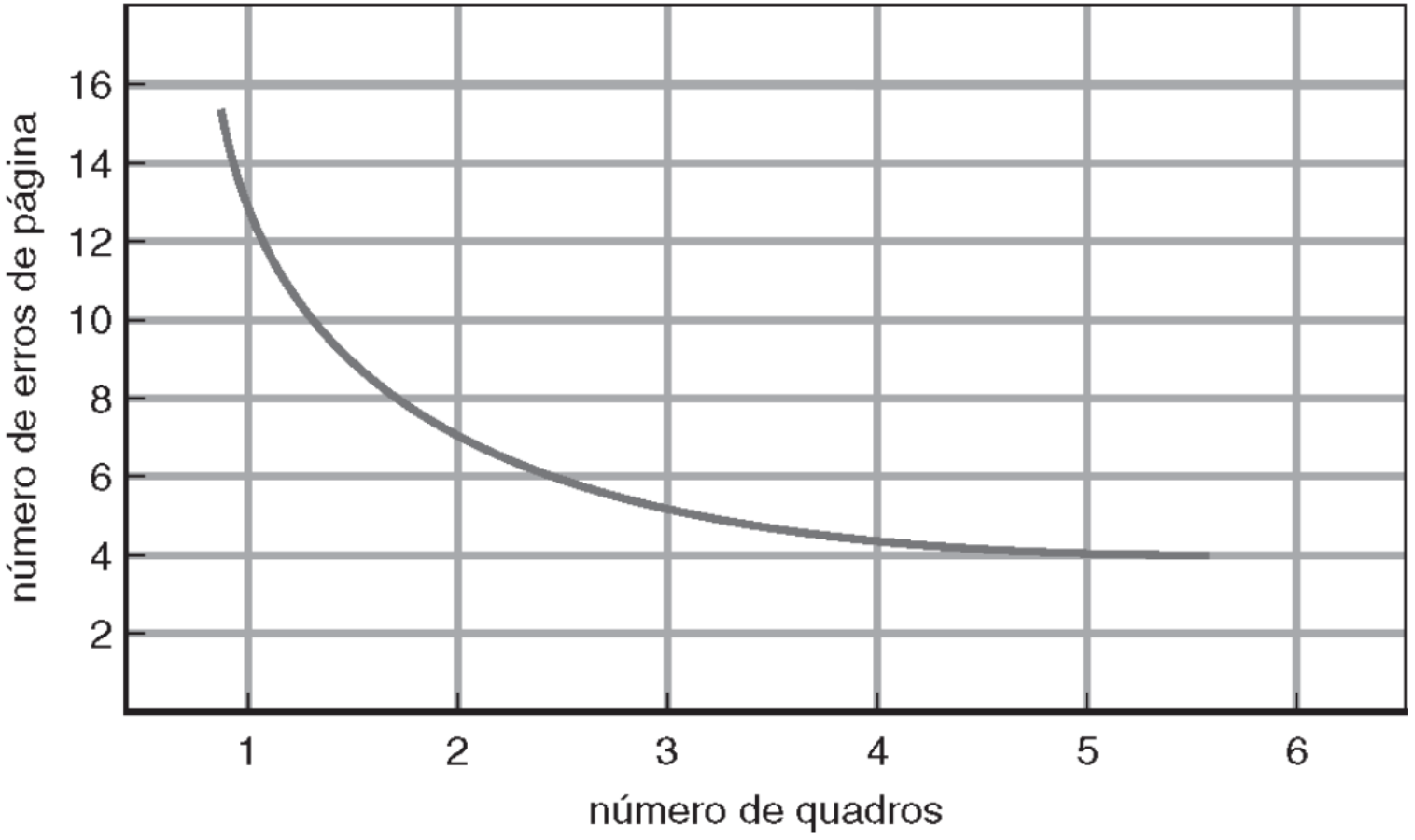
Algoritmos de substituição de páginas

- Queremos a menor taxa possível de faltas de página
- Avaliamos um algoritmo utilizando uma sequência de acessos a memória (sequência de referência) e contando a quantidade de faltas de páginas
- Em nossos exemplos utilizaremos as seguintes sequências:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

Gráfico de faltas de páginas versus número de quadros



First-In-First-Out (FIFO)

- Sequência de referência: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 quadros (3 páginas na memória ao mesmo tempo)

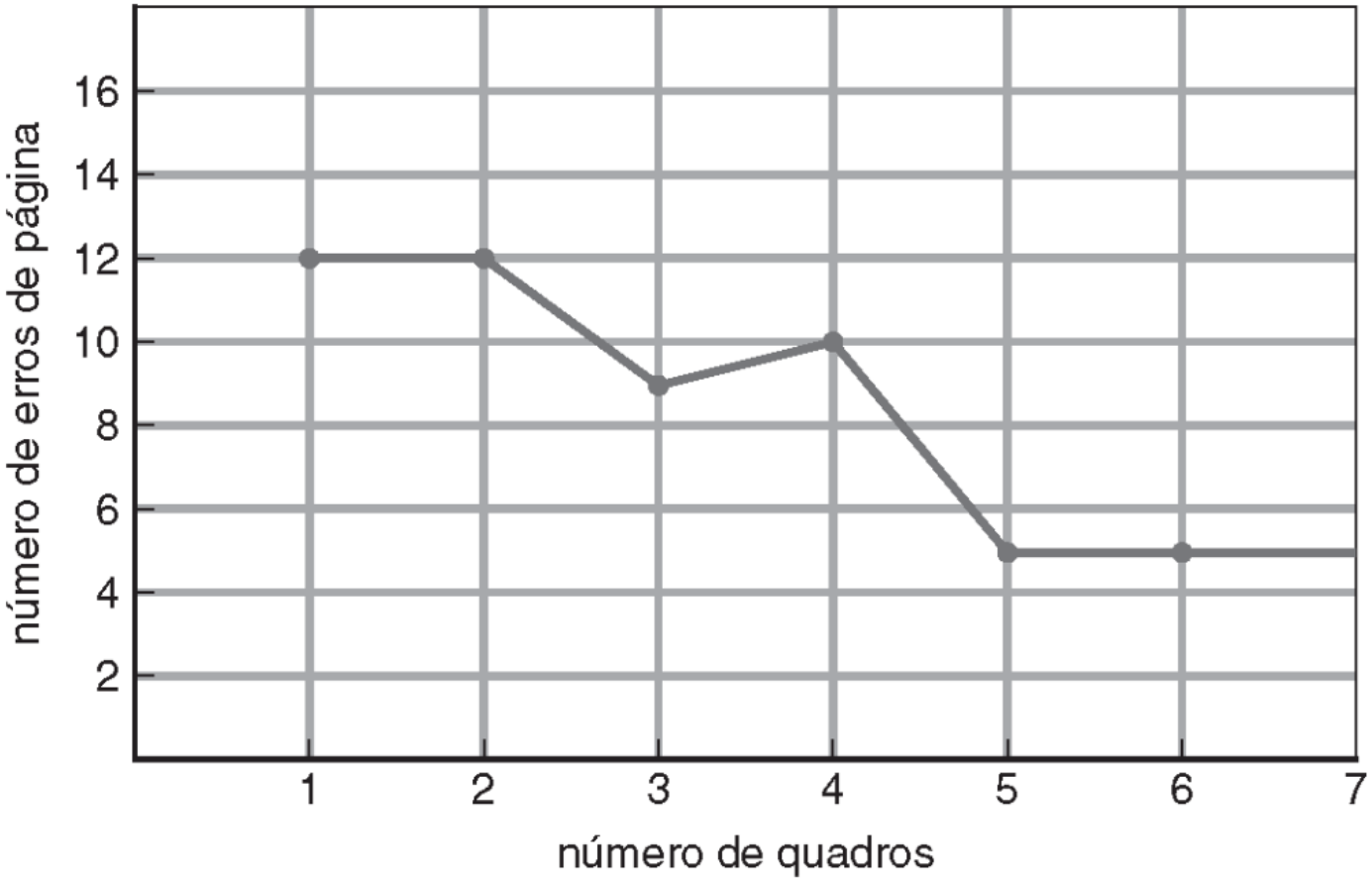
1	1	4	5	
2	2	1	3	9 faltas
3	3	2	4	

- 4 quadros

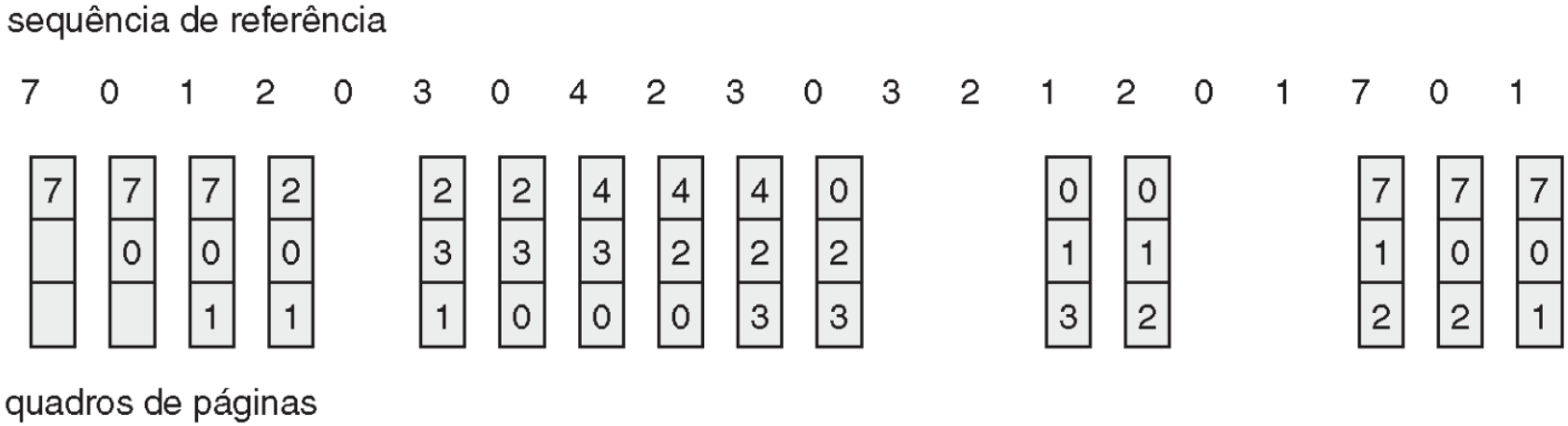
1	1	5	4	
2	2	1	5	10 faltas
3	3	2		
4	4	3		

- Anomalia de Belady: mais quadros → mais faltas de página

FIFO ilustrando a Anomalia de Belady



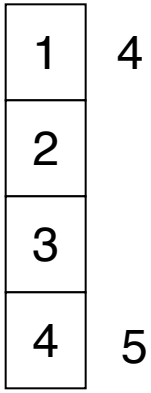
Substituição FIFO



Algoritmo Ótimo

- Substitui a página que não será acessada pelo maior período de tempo
- Exemplo com 4 quadros

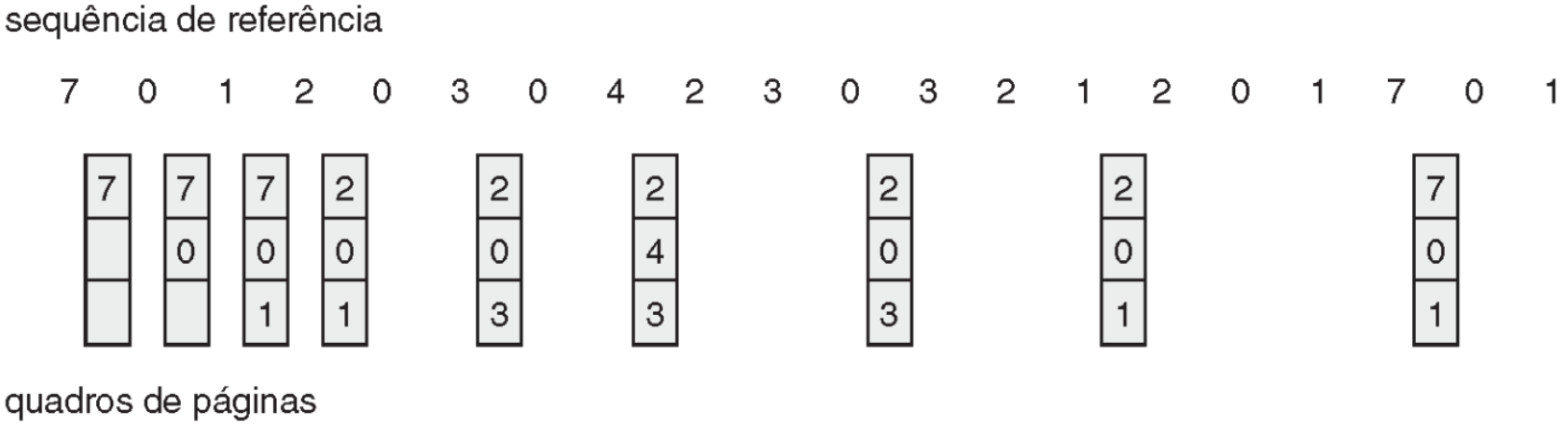
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



6 faltas

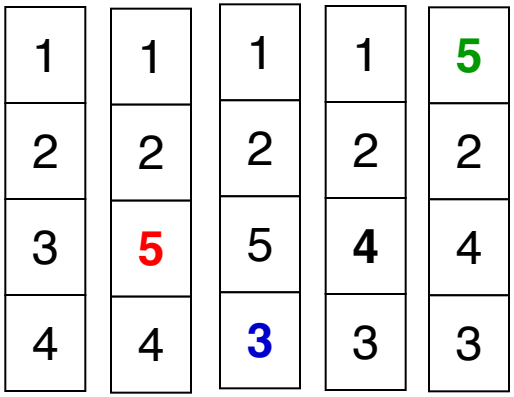
- Como obter essa informação?
- Útil para comparação com outros algoritmos

Algoritmo Ótimo

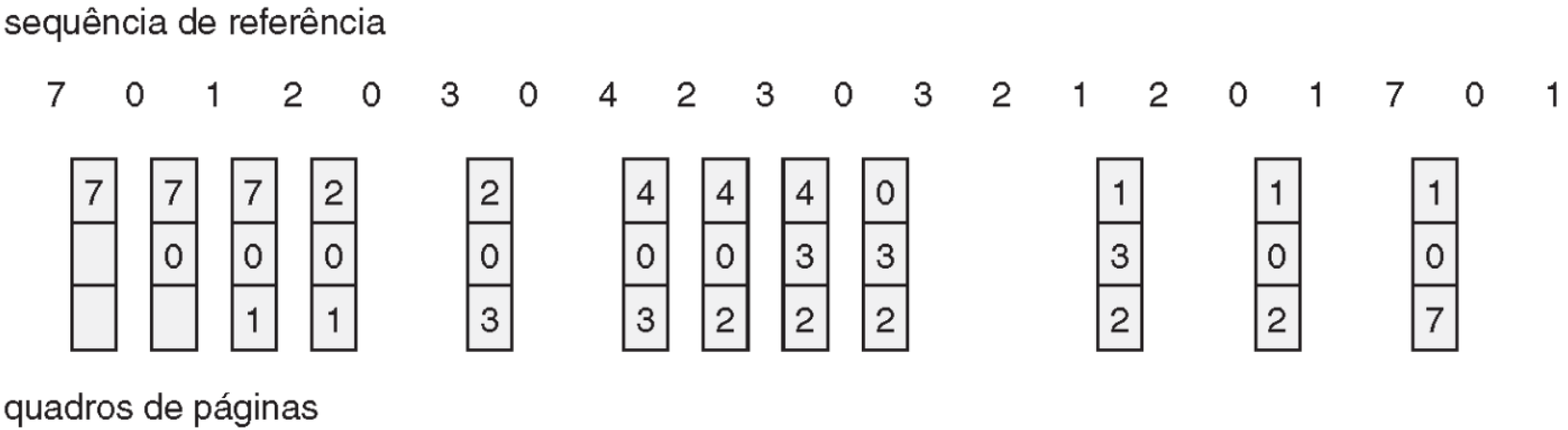


Menos Recentemente Usada (LRU)

- Sequência de referência: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**



Menos Recentemente Usada (LRU)



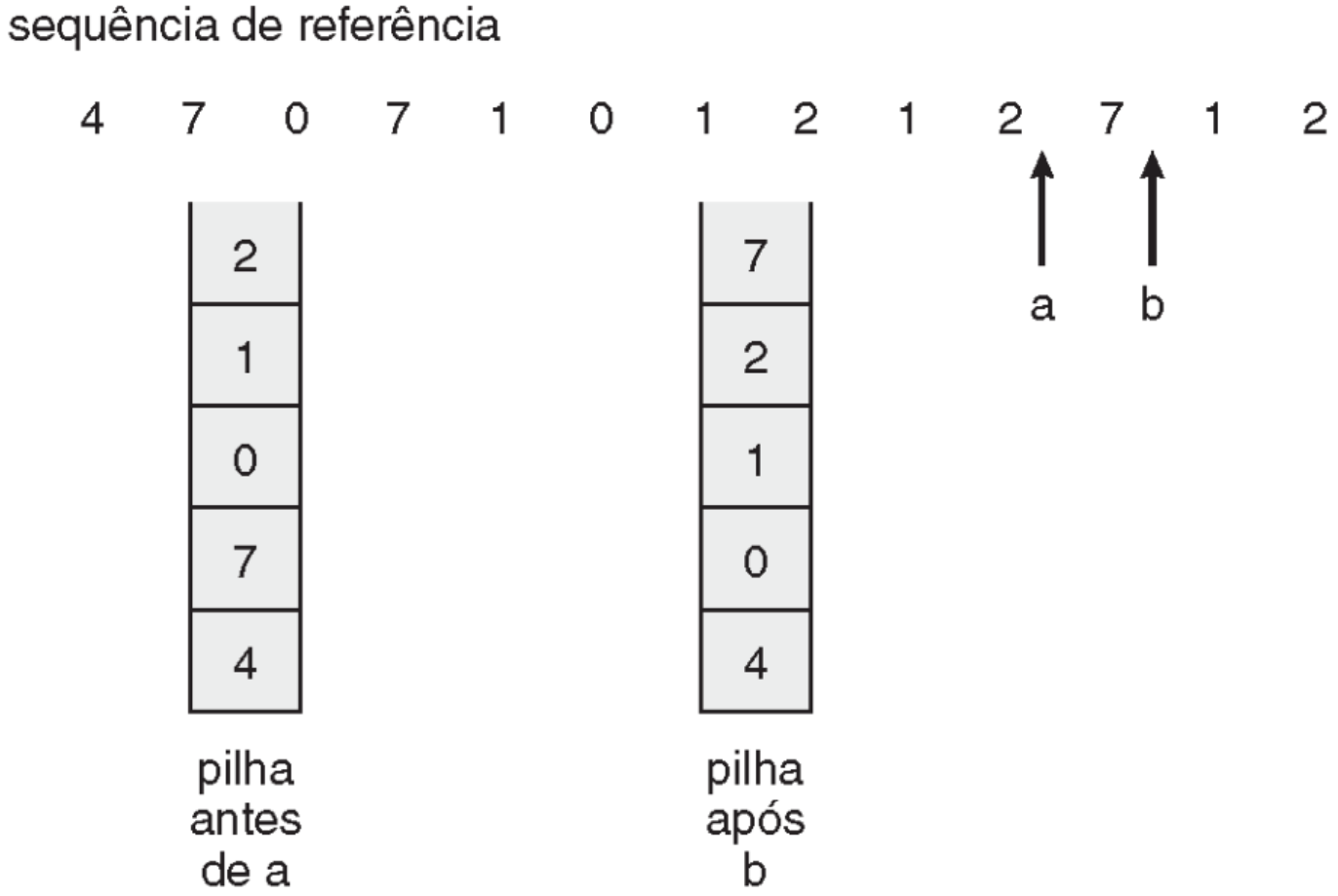
Menos Recentemente Usada (LRU)

- Implementação com contador
  - Cada página possui um contador que é atualizado com o valor do relógio sempre que a página é referenciada
  - Quando uma página precisa ser substituída, consultar o contador para remover a que possuir o menor valor
  - Desvantagens
    - Percorrer a tabela de páginas
    - Mecanismos para tratar estouro do relógio

Menos Recentemente Usada (LRU)

- Implementação com pilha – mantem uma pilha duplamente encadeada com os números das páginas:
  - Página referenciada
    - Move a página para o topo
  - Não requer algoritmo de busca
  - Desvantagem
    - Operações com ponteiros

LRU com a utilização de uma pilha



Aproximações do LRU

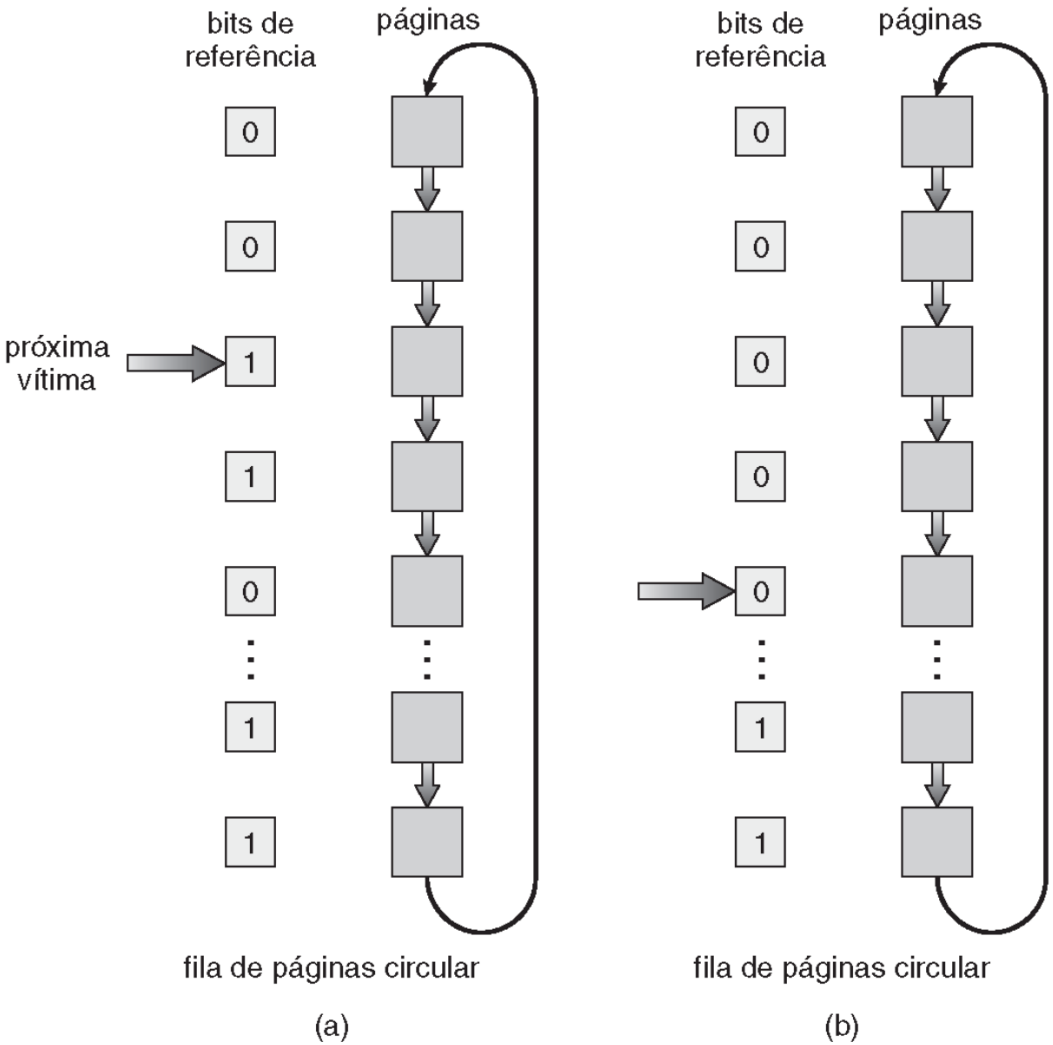
- Problema do LRU: suporte de hardware
  - Cada referência de memória para atualizar campo do relógio ou da pilha
  - Se fosse interrupção, sistema ficaria inviável

- Bit de referência
  - Associar um bit a cada página, inicialmente = 0
  - Quando a página é referenciada, bit = 1
  - Substitui a página com bit = 0 (se existir alguma)

Aproximações do LRU

- Segunda chance
  - Requer um bit de referência
  - Substituição do relógio
  - Se a página a ser substituída (em sentido horário) tem bit = 1, então
    - Seta bit = 0
    - Deixa a página na memória
    - Repete o processo para a próxima página (em sentido horário)

Algoritmo da Segunda Chance



Algoritmo de Contagem

- Mantem um contador para o número de referências a cada página
- **Algoritmo LFU**: substitui a página com o menor contador
  - Página usada com mais frequência tem maior contador
- **Algoritmo MFU**: substitui a página com o maior contador
  - Acabou de ser trazida para a memória

Alocação de Quadros

- Cada processo precisa de um número mínimo de quadros

- Dois esquemas principais de alocação
  - Alocação fixa
  - Alocação por prioridade

Alocação fixa

- Alocação igualitária: se há 100 quadros e 5 processos, alocar 20 quadros para cada um

- Alocação proporcional: quadros são alocados de acordo com o tamanho dos processos

$s_i =$ *tamanho* do processo p_i

$S = \sum s_i$

$m =$ número total de quadros

$a_i =$ alocação para $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_i = 10$

$s_2 = 127$

$a_1 = \frac{10}{137} \times 64 \approx 5$

$a_2 = \frac{127}{137} \times 64 \approx 59$

Alocação por prioridade

- Usa uma alocação proporcional baseada na prioridade ao invés do tamanho

- Se o processo P_i causa uma falta de página
  - Seleciona um de seus quadros para substituição
  - Seleciona um quadro de um processo de menor prioridade para substituição

Alocação global versus local

- **Substituição global:** o processo seleciona um quadro para substituição de uma lista com todos os quadros de memória - um processo pode tomar um quadro de outro

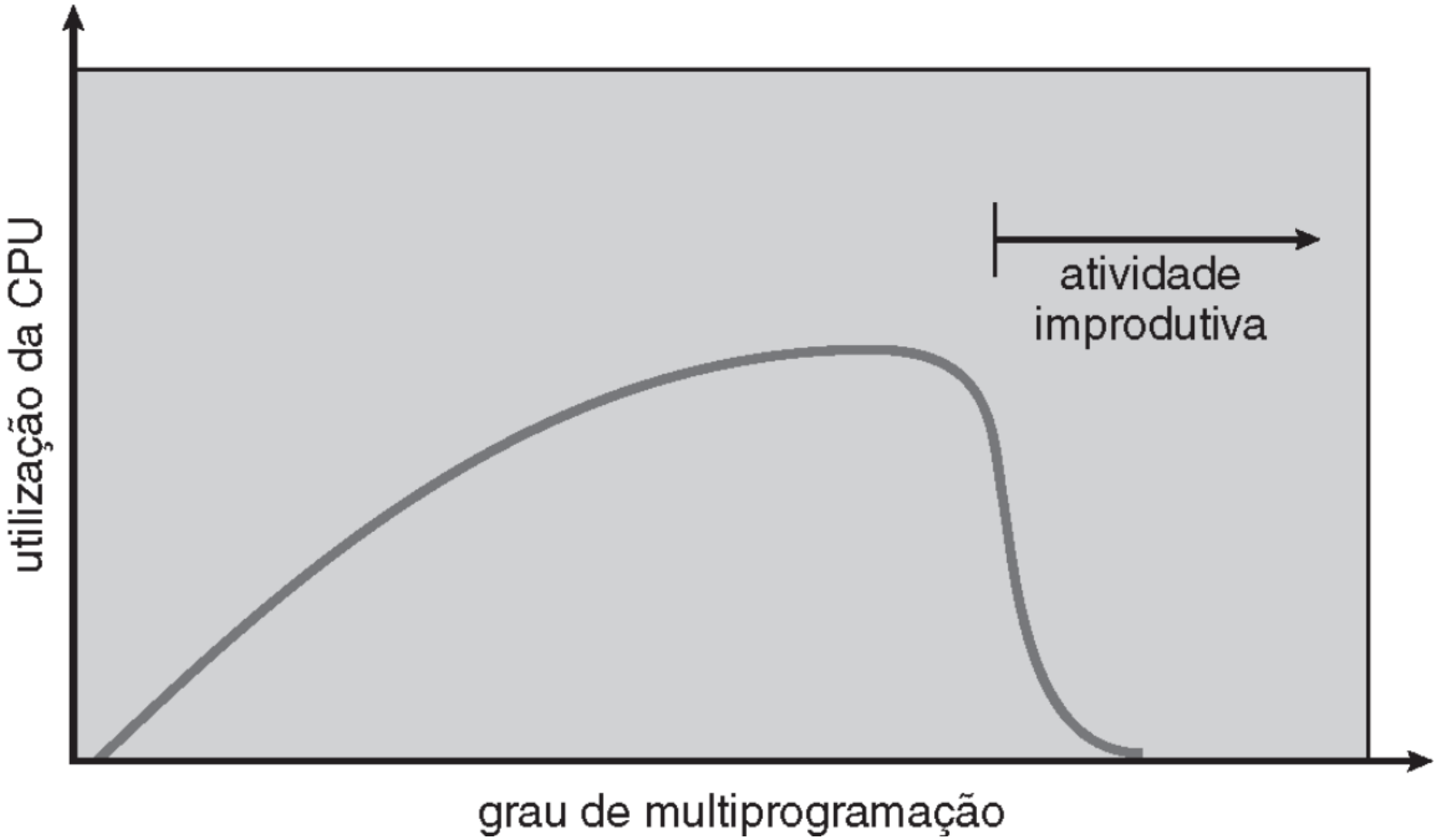
- **Substituição local:** cada processo só seleciona para substituição os seus próprios quadros

Thrashing

- **Thrashing** → um processo fica paralisado com operações de swap in e swap out

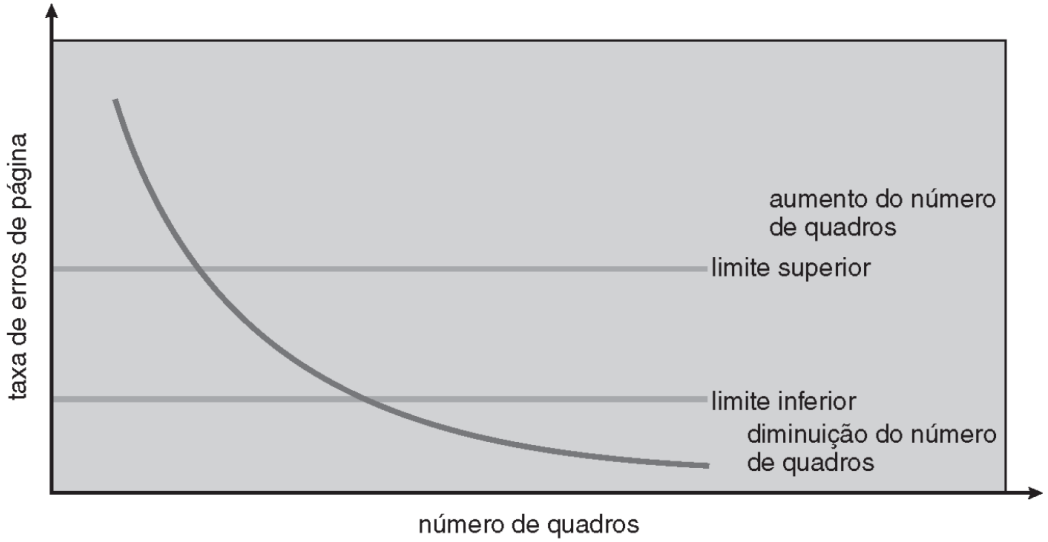
- Se um processo não tem quadros “suficientes” a taxa de falta de páginas pode ser muito alta. Isso leva a :
  - Baixa utilização da CPU
  - O Sistema Operacional conclui que precisa aumentar o grau de multi-programação
  - Um novo processo é adicionado à memória

Thrashing


utilização da CPU
atividade
improdutiva
grau de multiprogramação

Monitorando a taxa de faltas de página

- Definimos uma taxa aceitável de faltas de página
  - Se a taxa atual é muito baixa o processo perde um quadro
  - Se a taxa atual é muito alta o processo ganha um quadro



Exercícios

Para a sequência: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Quantas faltas de páginas com 5 quadros, os algoritmos LRU, Ótimo e FIFO geram.

Exercícios

Para a sequência: 1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Quantas faltas de páginas com 4 quadros, os algoritmos FIFO, Ótimo e LRU geram.

LRU - 10

OPT – 8

FIFO - 14