BC1518 - Sistemas Operacionais

Aula 9: Memória Virtual

Material parcialmente baseado nos slides do Prof. José Artur Quilici Gonzalez

Tópicos desta aula

- Fundamentos
- Paginação Sob Demanda
- Substituição de Páginas - Algoritmos
- Alocação de Quadros
- *Thrashing*

Fundamentos

➢Memória Virtual

- Técnica de gerência de memória que combina as memórias principal (RAM) e secundária (disco)
- Somente parte do programa precisa estar na memória física para a execução
  - Os programas podem, portanto, ser maiores do que a memória física
  - Maior compartilhamento de memória pelos processos
  - Reduz os problemas de fragmentação de memória
- Permite que espaços de endereçamento sejam compartilhados por vários processos

Espaço de endereçamento virtual

➢Um espaço de endereçamento é uma abstração para a memória

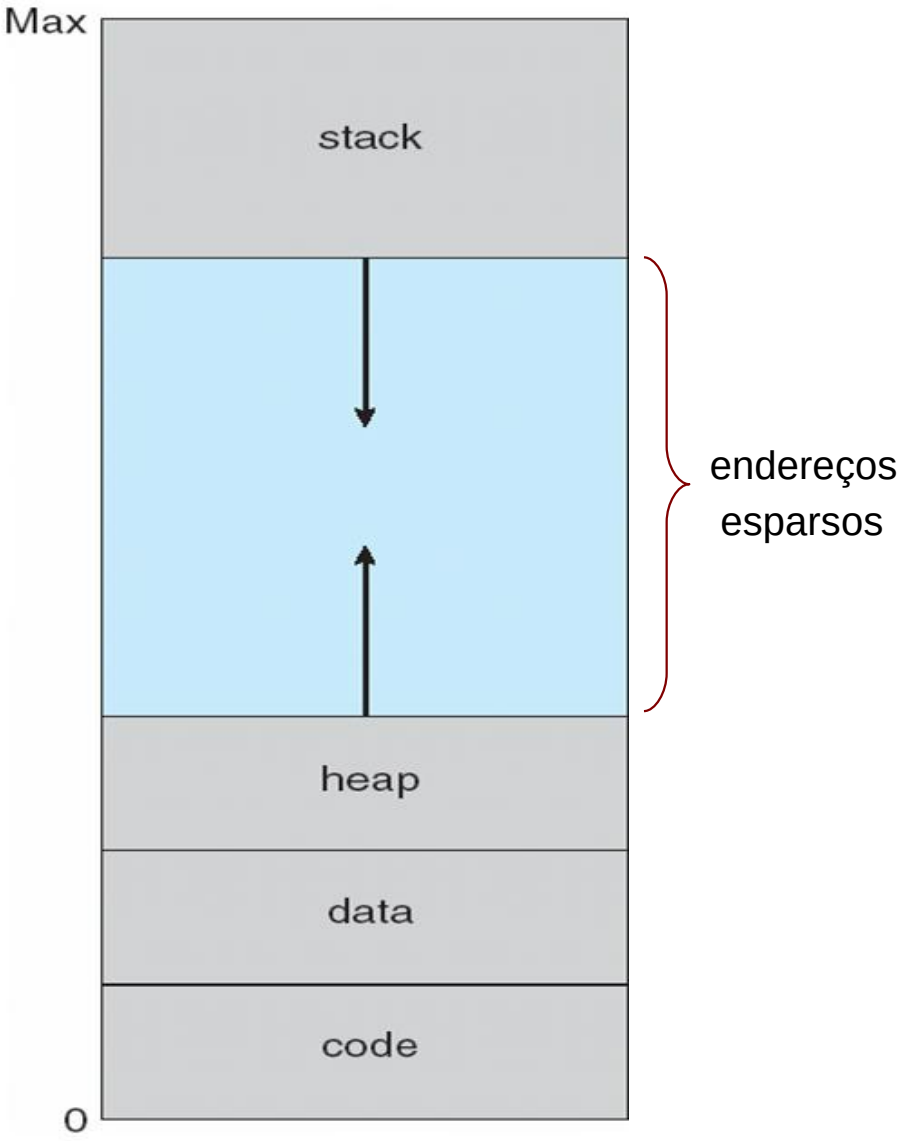
- Permite a separação entre a memória lógica (vista pelos usuários) e a memória física (real)
  - Possibilita uma memória lógica >> memória física real

- Os processos não fazem referência direta a endereços físicos de memória somente a endereços virtuais

- O espaço de endereçamento virtual é o conjunto de endereços virtuais que o processo pode referenciar

- De forma análoga, o espaço de endeçamento real é o conjunto de endereços reais (físicos) que o processador pode referenciar

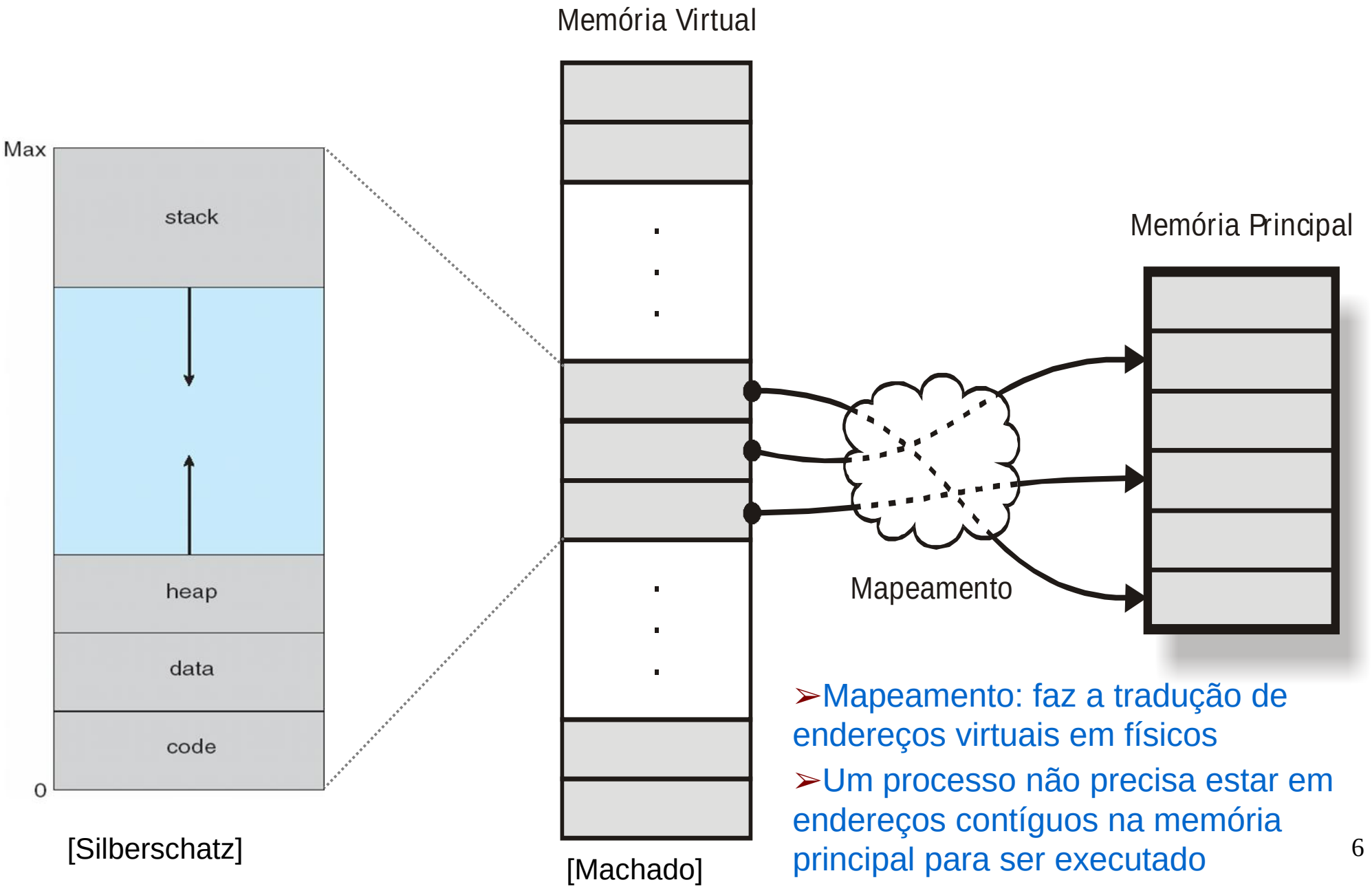
Espaço de endereçamento virtual



[Silberschatz]

- O espaço de endereços virtuais de um processo: refere-se à visão lógica (ou virtual) de como um processo é armazenado na memória
  - um processo começa em um certo endereço lógico (ex.: endereço 0) e está em trechos contíguos de memória
  - A pilha e *heap* podem crescer dinamicamente, os espaços vazios fazem parte do espaço de endereçamento virtual mas só só exigirá páginas físicas reais (memória) se a pilha ou *heap* crescer

Mapeamento



[Silberschatz]

[Machado]

- Mapeamento: faz a tradução de endereços virtuais em físicos
- Um processo não precisa estar em endereços contíguos na memória principal para ser executado

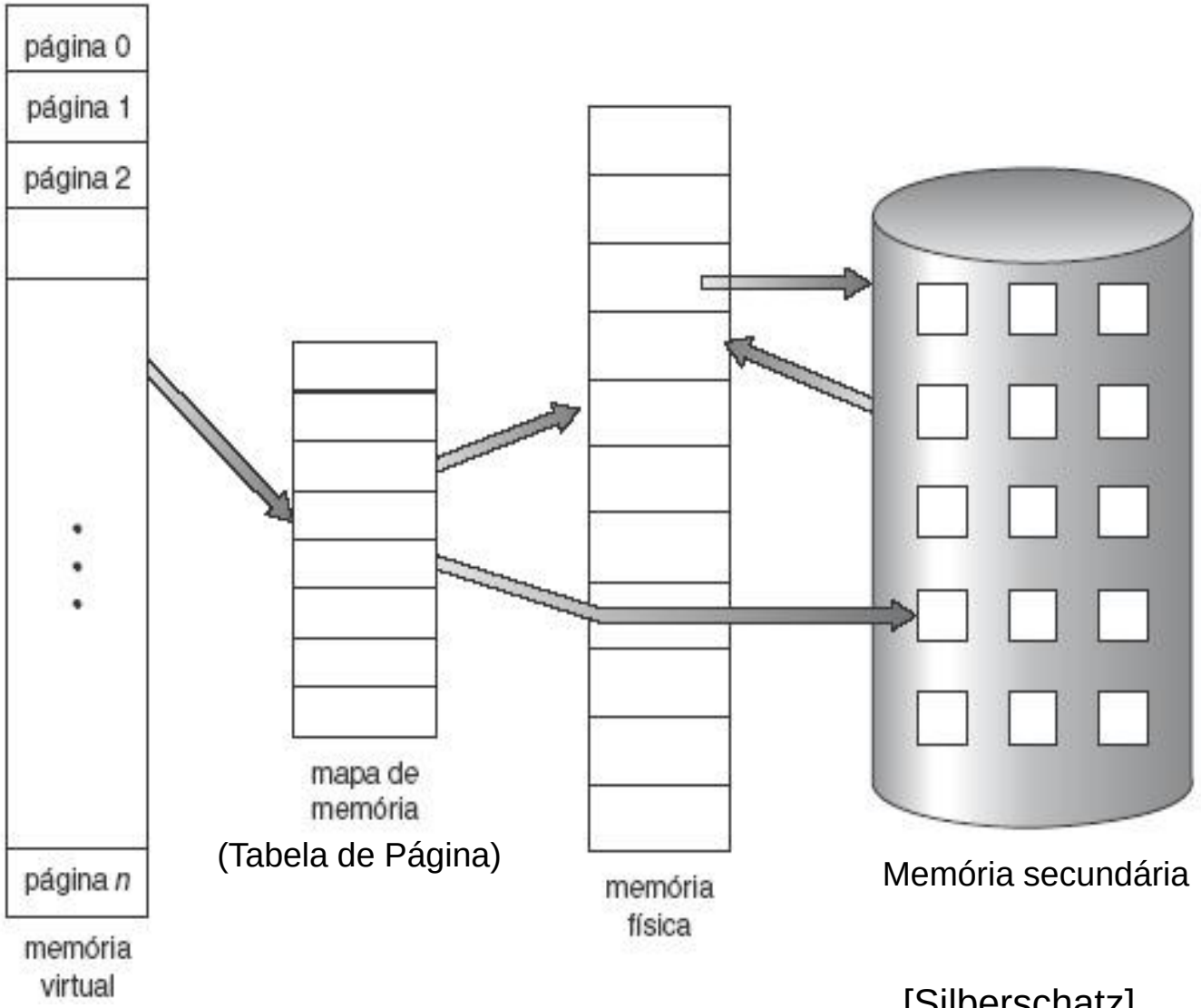
Paginação

- Esquema de gerência de memória que permite que o **espaço de endereçamento físico** de um processo possa ser **não-contíguo**; o processo é alocado na memória física onde houver blocos livres

- Espaço de endereços virtuais é dividido em blocos: páginas virtuais
- Espaço de endereços reais também é dividido blocos de mesmo tamanho das páginas virtuais: páginas reais ou quadros (*frames)*
- O mapeamento de endereço virtual em real é feito através de Tabela de Páginas
  - Cada processo possui sua própria tabela de páginas

- A tarefa de conversão de endereços virtuais em físicos é feito pela MMU *(Memory Management Unit)*

Memória virtual



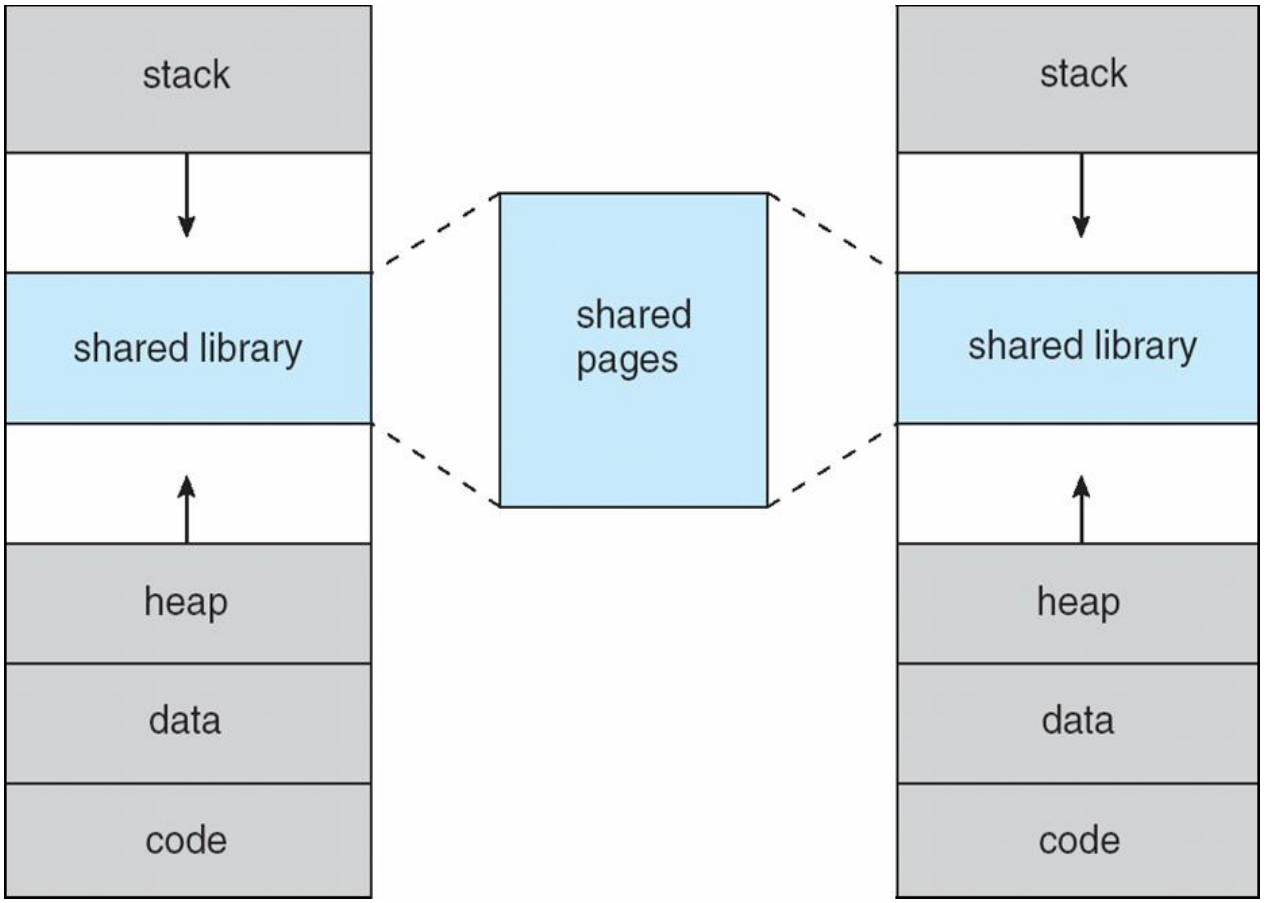
[Silberschatz]

- A memória virtual pode ser muito maior do que a memória física, o disco é uma extensão da memória física
- Se uma **página** solicitada não estiver carregada na **memória física**, ela deve ser trazida do **disco rígido** (e possivelmente outra página já utilizada deva ser levada de volta ao disco rígido)

Compartilhamento de memória

- A memória virtual permite o compartilhamento de arquivos (código) ou memória por dois ou mais processos diferentes (através do compartilhamento de página)
  - Compartilhamento de bibliotecas do sistema: cada processo vê as bibliotecas como parte de seu espaço de endereçamento virtual mas as páginas reais em memória física são compartilhadas por todos os processos → as entradas na tabela de página de cada processo contém os mesmos quadros da memória principal
  - Compartilhamento de memória pelos processos: comunicação por memória compartilhada
  - Também permite que páginas sejam compartilhadas durante a criação do processo (utilizando a chamada de sistema fork()), agilizando a criação do processo

Biblioteca compartilhada usando memória virtual



[Silberschatz]

A **Memória Virtual** permite que diferentes processos compartilhem a mesma região da **Memória Física**

Memória virtual

- Memória Virtual pode ser implementada por:
  - **Paginação Sob Demanda** (mais comum)
    - Blocos (páginas) de tamanho fixo
  - Segmentação Sob Demanda (mais complexa – OS/2, Burroughs)
    - Blocos (páginas) de tamanho variável

Paginação sob demanda

➢Um programa pode ser carregado totalmente na memória mas muitas vezes não é preciso dele inteiro inicialmente para execução

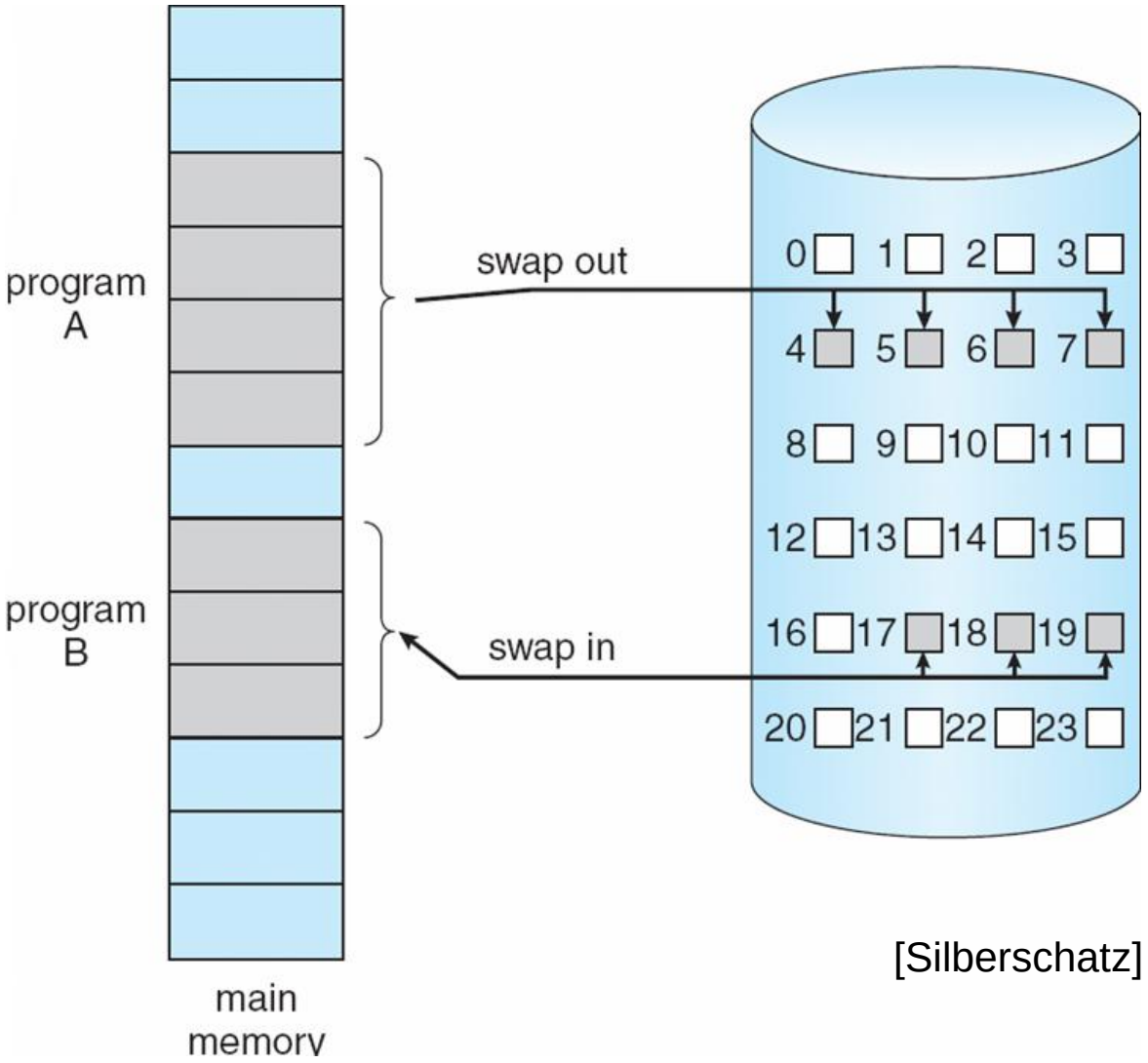
➢Paginação sob demanda: carrega uma página na memória apenas quando esta é necessária

- Necessita menos E/S
- Necessita menos memória física, pois apenas parte do processo é carregado
- Possibilita um maior número de processos na memória

➢Se uma página é necessária → basta referenciá-la

- Se a página não está na memória → basta carregá-la na memória

Transferência de uma memória paginada para espaço de disco contíguo



[Silberschatz]

Um sistema de paginação sob demanda é semelhante a um sistema de paginação com *swapping*, porém sem a necessidade de carregar todas as páginas ao mesmo tempo

Como identificar se uma página está ou não na memória?

➢Para identificar se uma página está ou não na memória:

- A cada entrada na **tabela de página** é associado um *bit* válido – inválido:

1 → está na memória, 0 → não está na memória

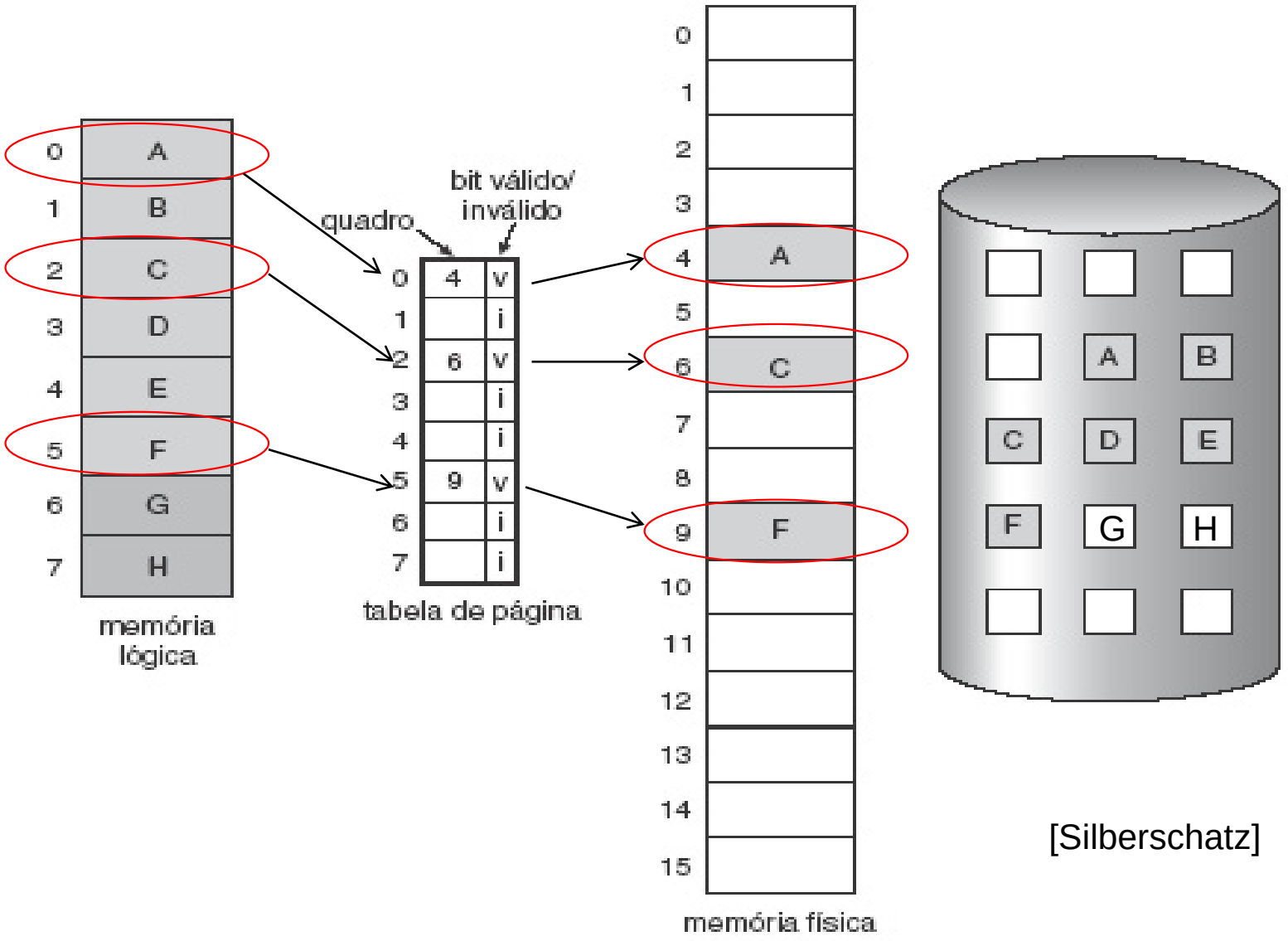
- Inicialmente todos os *bits* estão em 0
- Exemplo de tabela de página:

Quadro #	Bit válido-inválido
	1
	1
	1
	1
	0
	0
	0

Tabela de página

➢Durante a tradução a um endereço, se o *bit* válido–inválido na tabela de página for 0 → ocorre falha de página

Bit válido - inválido



[Silberschatz]

➢No exemplo, as páginas A, C e F estão na memória (páginas *residentes* na memória) e estão marcadas como válidas; As outras páginas estão como inválidas, ou seja, não estão na memória

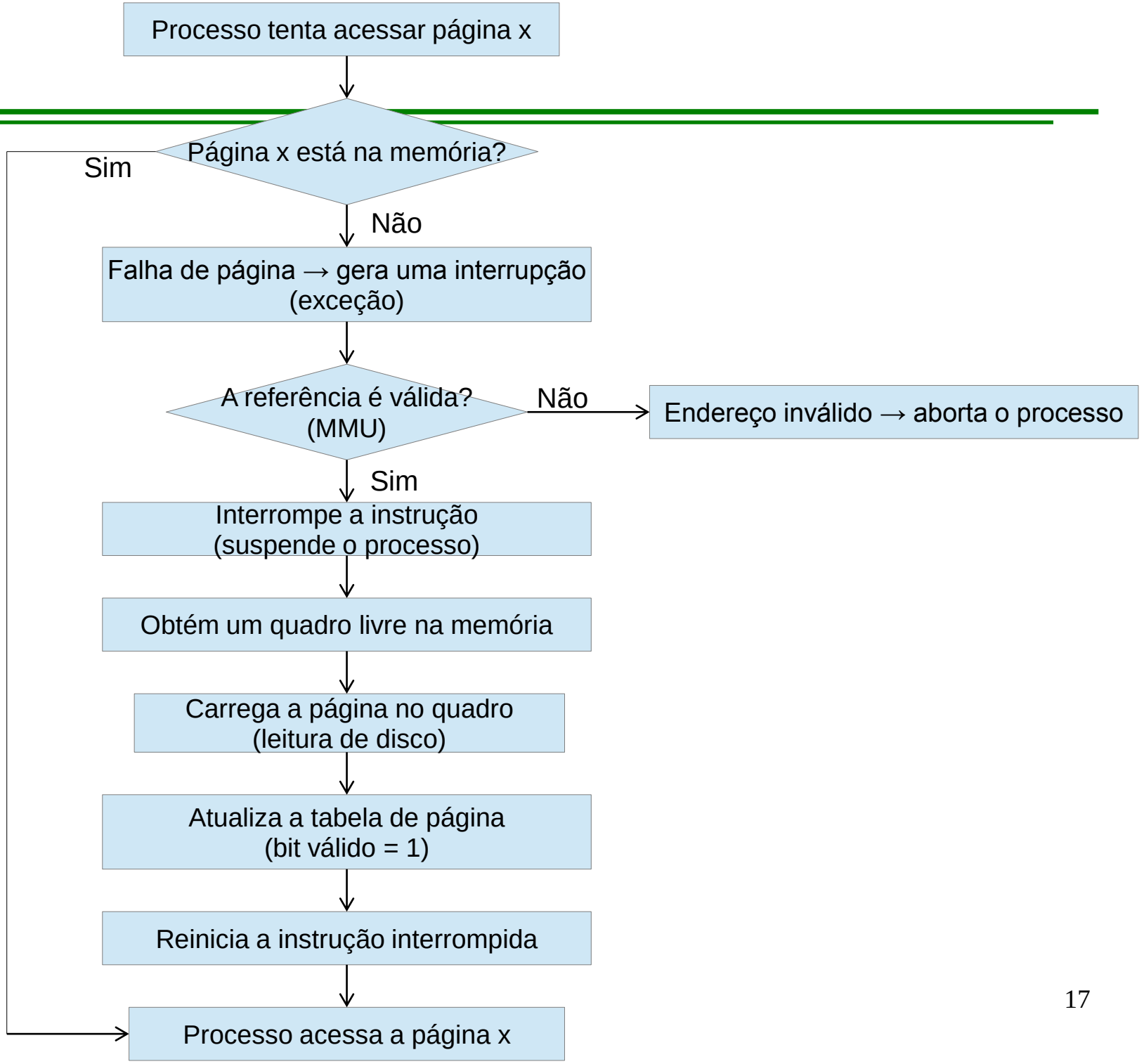
Falta (ou falha) de página

➢O que acontece se o processo tentar acessar uma página que não está na memória?

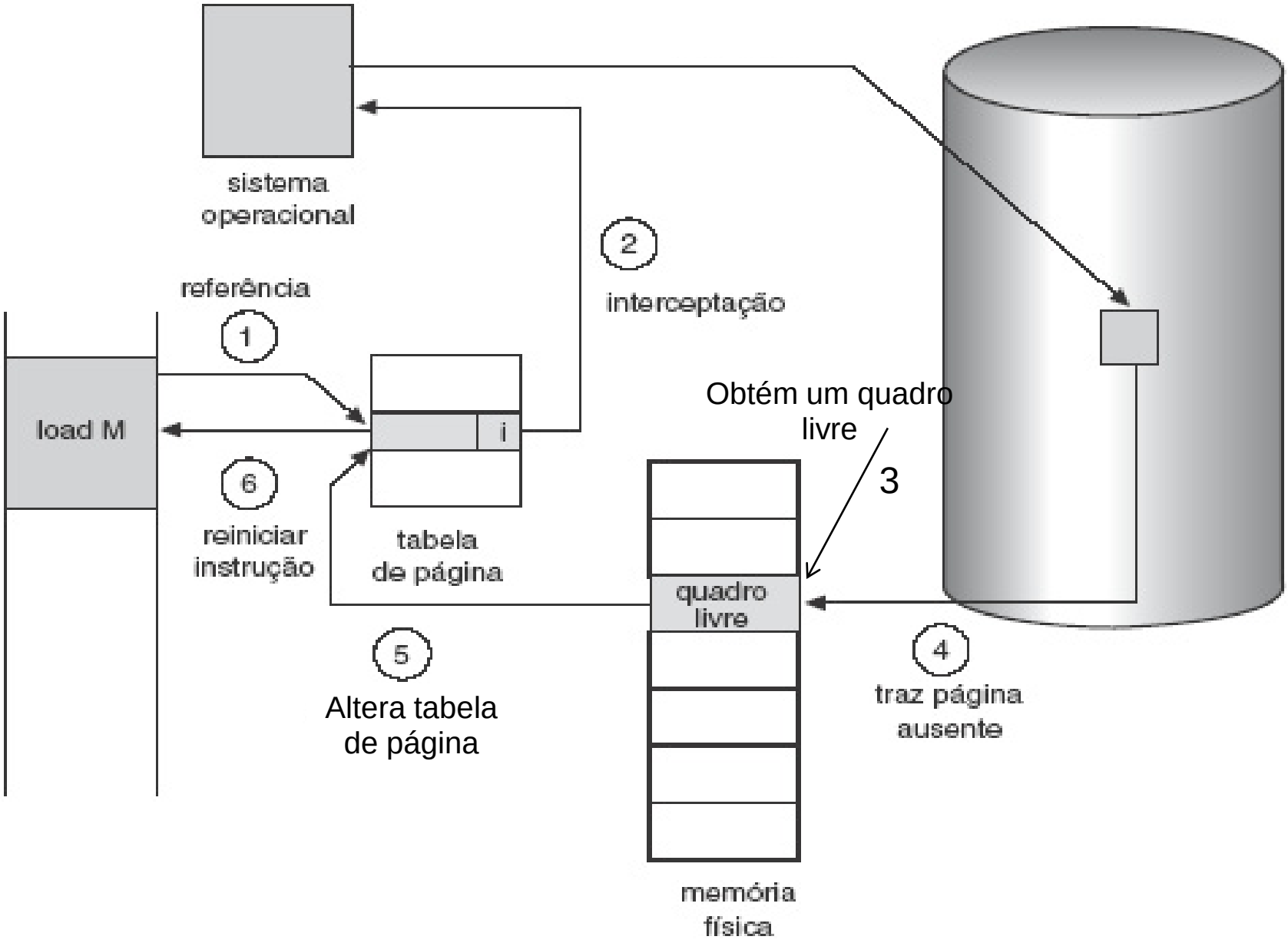
- O acesso a uma página marcada como inválida (não está na memória) → gera uma interrupção ou interceptação por falha de página *(page fault trap)* → falta ou falha de página
- O hardware de paginação (MMU) ao traduzir o endereço virtual na tabela de página identifica o bit marcado como inválido gerando a interrupção para o SO

➢Para tratar a falha de página:

- O SO consulta uma tabela interna (geralmente mantida com o PCB do Processo) para decidir se:
  - Referência inválida → *abort* (termina o processo)
  - Válida, porém a página não está carregada
  - Obtém um quadro livre
  - Carrega a página desejada no quadro (operação de E/S)
  - Atualiza a tabela, ajustando-se o *bit* válido = 1
  - Reinicia a instrução interrompida


Processo tenta acessar página x
Página x está na memória?
Sim
Não
Falha de página → gera uma interrupção (exceção)
A referência é válida? (MMU)
Não
Endereço inválido → aborta o processo
Sim
Interrompe a instrução (suspende o processo)
Obtém um quadro livre na memória
Carrega a página no quadro (leitura de disco)
Atualiza a tabela de página (bit válido = 1)
Reinicia a instrução interrompida
Processo acessa a página x

Etapas do tratamento de falta de páginas



O que acontece se não há quadro livre?

- **Substituição de página:** escolher uma página na memória que não esteja em uso e armazená-la no disco
  - A página escolhida (conteúdo) é transferido para o disco e a tabela de página é alterada de acordo, indicando que o processo não está mais na memória
  - A página requisitada é trazida do disco para a memória
  - Qual algoritmo usar?
  - *performance* (desempenho) – é desejável um algoritmo que produza um número mínimo de faltas de páginas

- Uma mesma página pode ser carregada/retirada da memória várias vezes durante a execução do processo

Esquema básico de substituição

Quando ocorre uma falha de página: é necessário carregar a página requerida do disco para a memória:

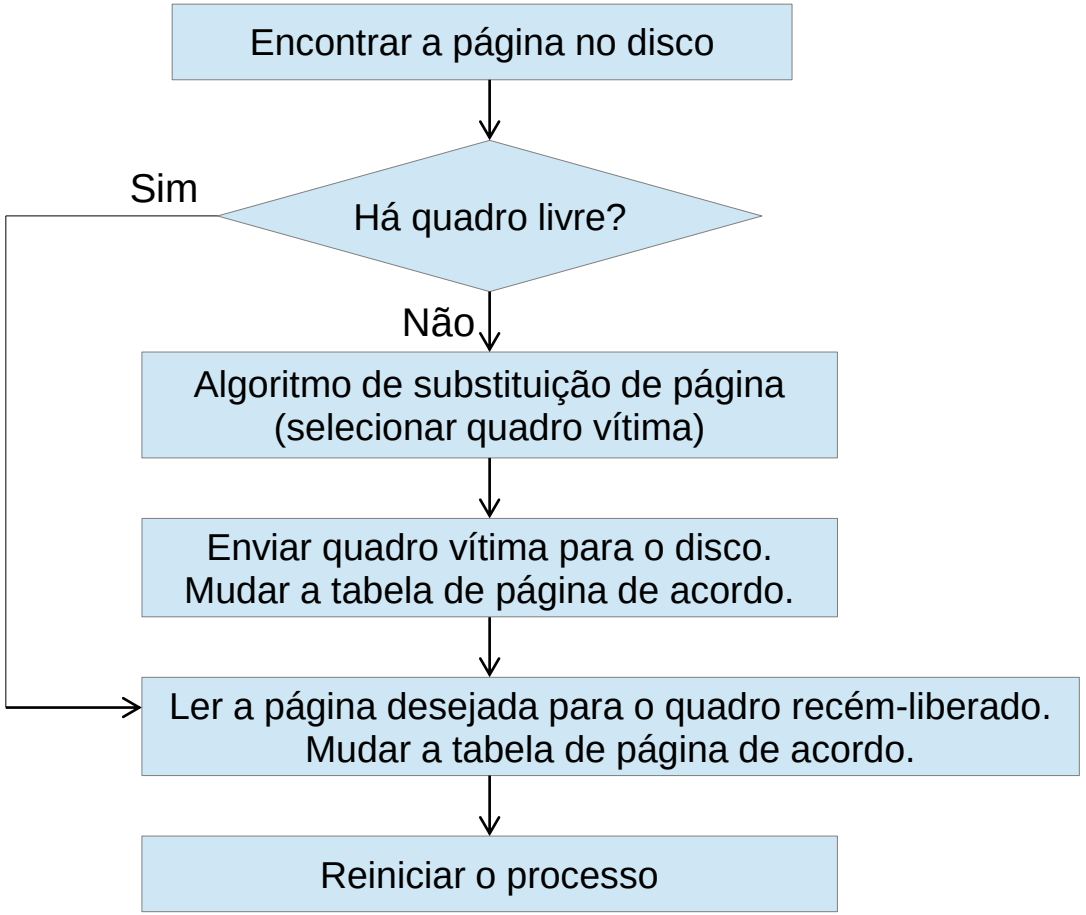
1. Encontrar a localização da página requerida no disco

2. Encontrar um quadro livre:
   - Se houver um quadro livre, use-o
   - Se não houver quadros livres, use um algoritmo de substituição de página para selecionar um quadro vítima (ou *página* vítima)
   - Transferir o quadro vítima para o disco
3. - Alterar a **tabela de página** e lista de **quadros livres** de acordo

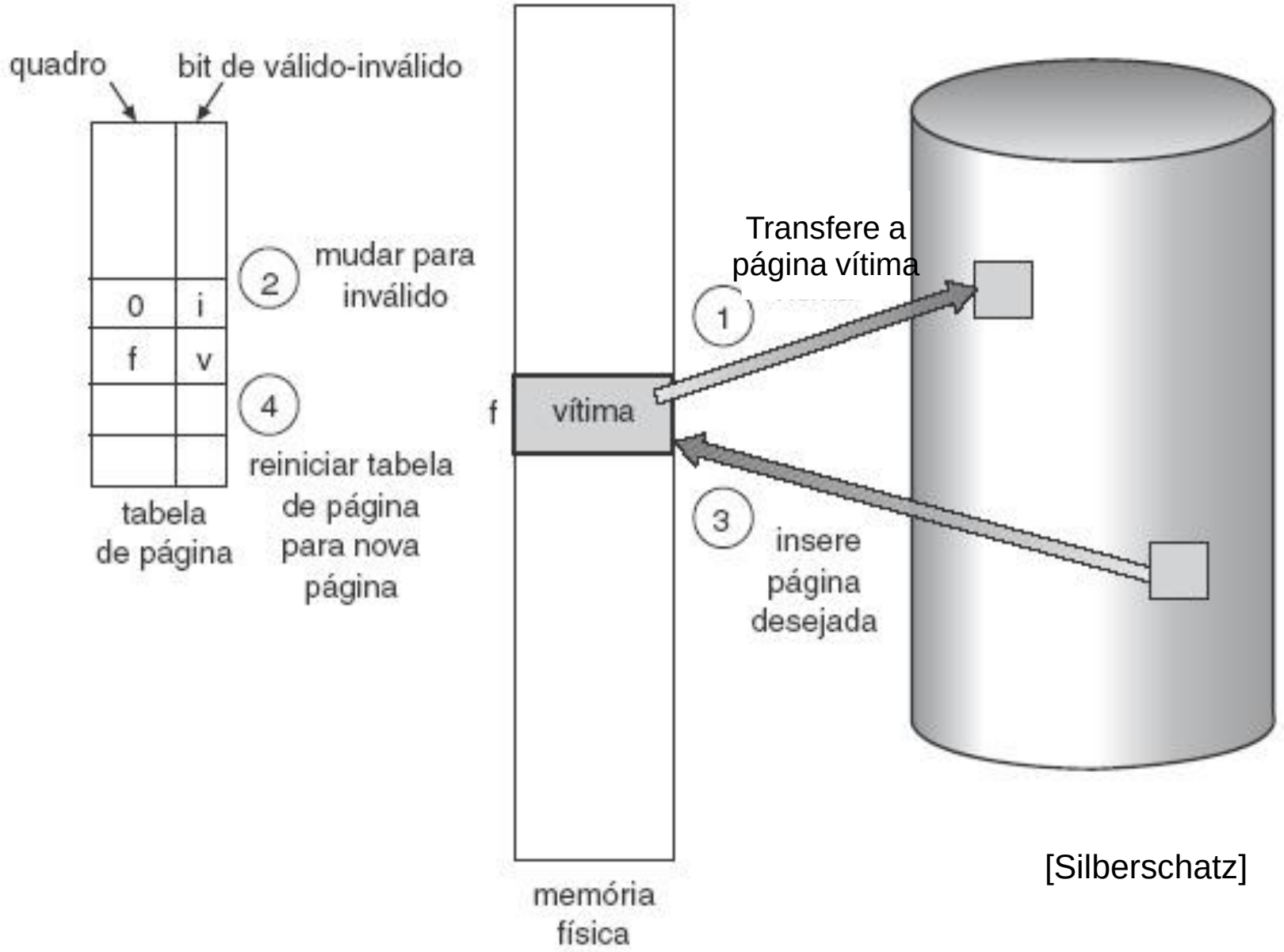
4. Carregar a página desejada no quadro (recém- liberado); atualizar a **tabela de página** e a lista **de quadros livre**

5. Reiniciar o processo

Esquema básico de substituição

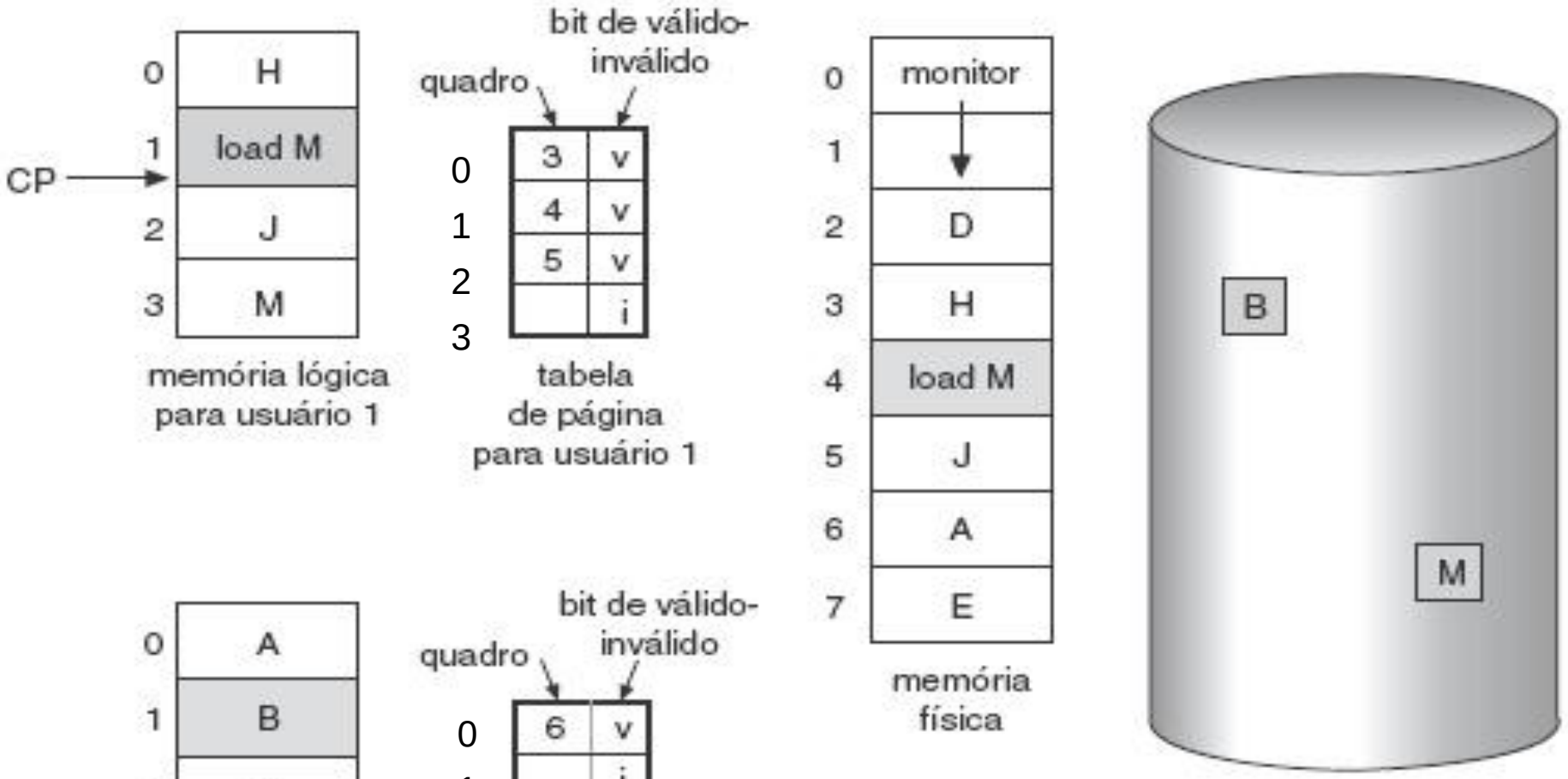


Substituição de página (cont.)



[Silberschatz]

Necessidade de substituição de página



[Silberschatz]

Inicialmente todas as páginas do **usuário 2** estavam carregadas, mas por falta de espaço para rodar o processo do **usuário 1**, a página 1 (B) do **usuário 2** foi devolvida ao disco para que a página 1 do **usuário 1** (M) pudesse ser carregada

Substituição de página

➢Quando é preciso fazer uma substituição de página (não há quadro livre) são necessárias duas transferências de páginas (uma para fora e outra para dentro da memória)

- Isso dobra o tempo para tratar uma falha de página, aumentando o tempo de acesso efetivo à página

➢Usa *bit* de modificação (*dirty bit*) para reduzir o custo de transferências de páginas – somente as páginas modificadas são gravadas no disco

- O bit de modificação é associado a cada página na tabela de página e é marcado sempre que houver modificação na página (escrita na memória)
- Quando uma página é selecionada para substituição, verifica se o bit de modificação está marcado e em caso afirmativo a página deve ser transferida para o disco.
- Porém, se não estiver marcado, a página não foi modificada desde que foi trazida para a memória, e portanto não será necessário

Algoritmos de substituição de páginas

➢Algortimo de substituição de página: decide qual quadro da memória deve ser enviado para disco para liberar espaço

- O objetivo é selecionar quadros que tenham as menores chances de serem referenciados num futuro próximo → evitando-se novas falhas de página
- A seleção de um algoritmo é muito importante pois operação de E/S em disco é dispendiosa

- Como selecionamos um algoritmo de substituição de páginas?

- Busca-se o algoritmo com a **menor taxa de falta de página**

➢Um algoritmo é avaliado executando-o sobre uma determinada série de referências de memória (***string* de referência**) e calculando o número de falhas de página

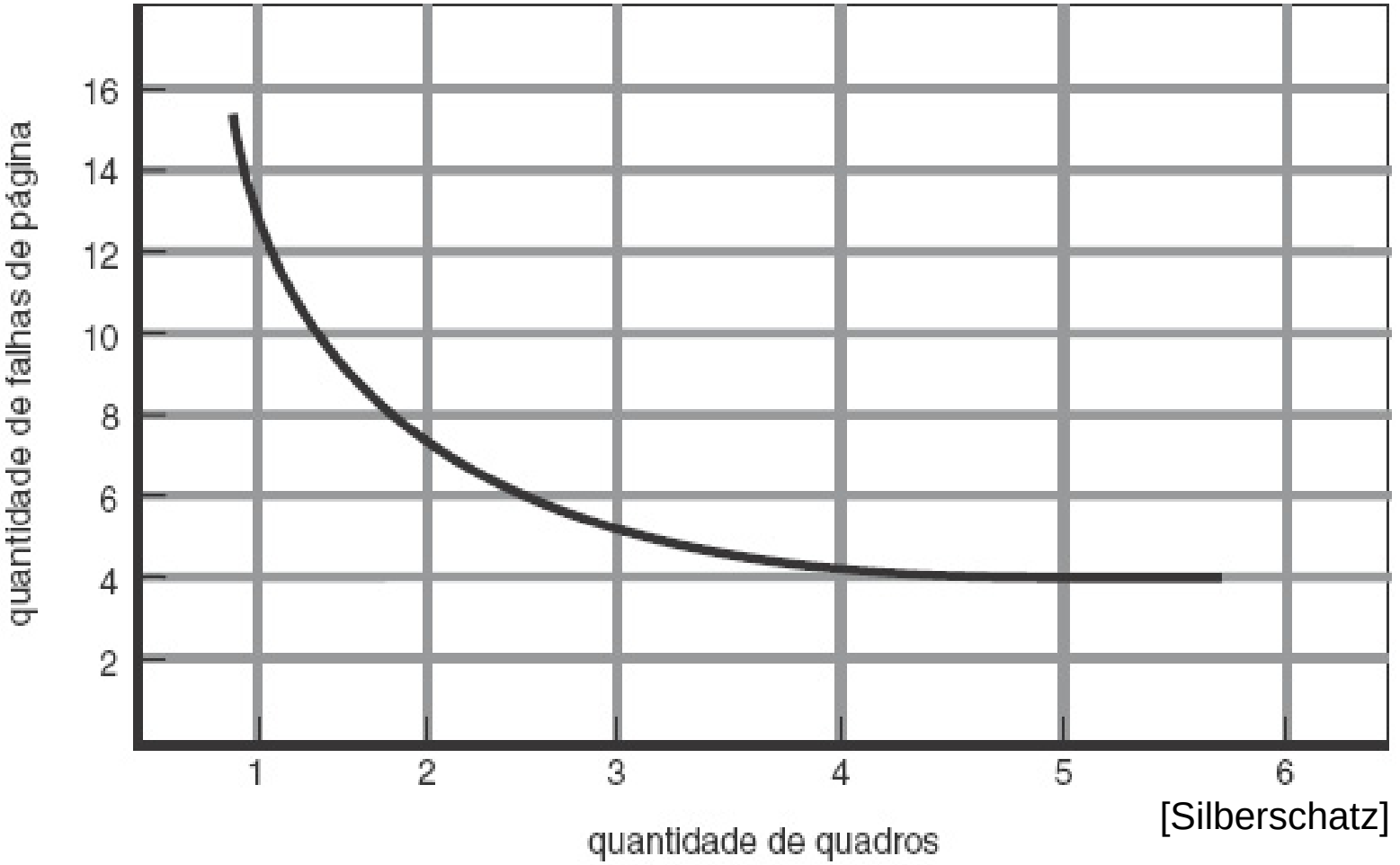
Algoritmos de substituição de páginas

- String de referência: conjunto de referências à memória
  - Uma sequência de endereços por ser algo como:
  - 100, 432, 101, 102, 230, 520…
  - Para simplificar, se cada página tiver 100 linhas, as referências 100, 101 e 102 estão na mesma página (página 1) e essa sequência é reduzida para a seguinte string de referência:
  - 1, 4, 1, 2, 5
  - Na string de referência usa-se o número da página
  - Apenas a primeira referência pode causar falha de página

- Nos exemplos a seguir, a *string* de referência utilizada é:
- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

- Para a avaliação dos algoritmos de substituição de páginas é preciso considerar também o número de quadros

Gráfico de falha de página vs número de quadros



[Silberschatz]

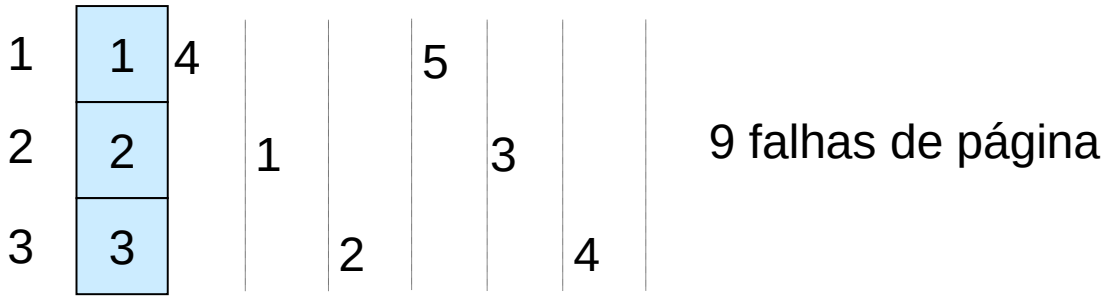
Em geral, o número de falhas de página diminui quando aumenta-se o número de quadros livres alocados ao processo

Algoritmo First-In-First-Out (FIFO)

- Algoritmo mais simples de substituição de páginas: FIFO (está relacionado com o tempo em que a página foi trazida para a memória, quando é preciso substituição de página, a mais antiga é escolhida)
- Mantém uma fila FIFO com todas as páginas trazidas para a memória (escolhe a primeira da fila)
- Quando uma nova página é trazida para a memória, entra para o fim da fila

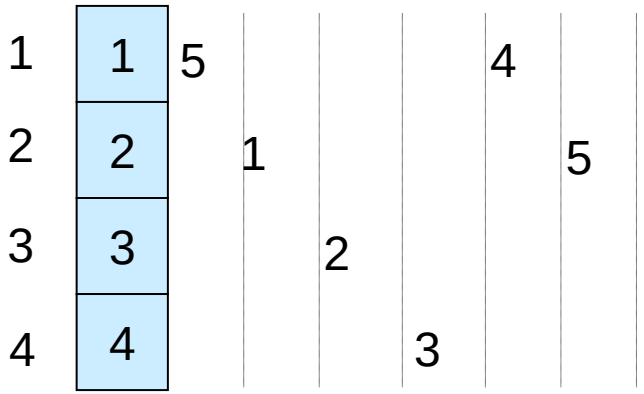
String de referência: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 (referências para as páginas)

3 quadros (inicialmente vazios) → somente 3 páginas na memória



Algoritmo First-In-First-Out (FIFO)

Com 4 quadros:



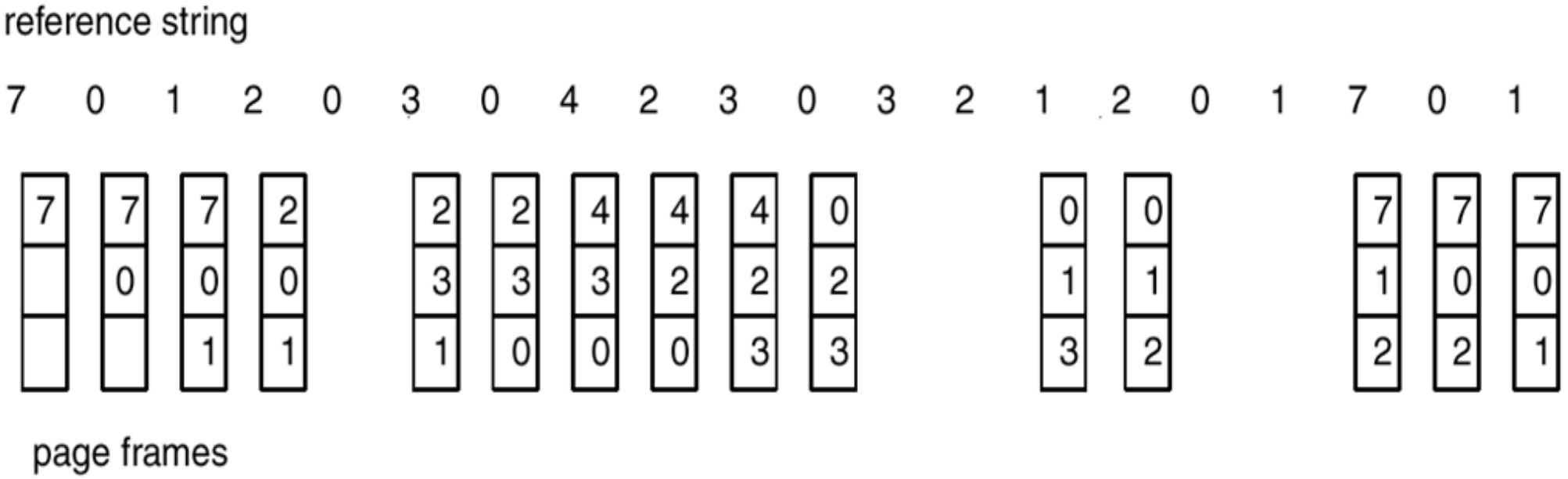
10 falhas de página

➢ Neste caso, com 4 páginas houve mais falhas de página

➢ Substituição FIFO – Anomalia de Belady

Há situações excepcionais em que quando há mais quadros → ocorrem mais falhas de páginas (quando o esperado seria o contrário)

Substituição de página FIFO

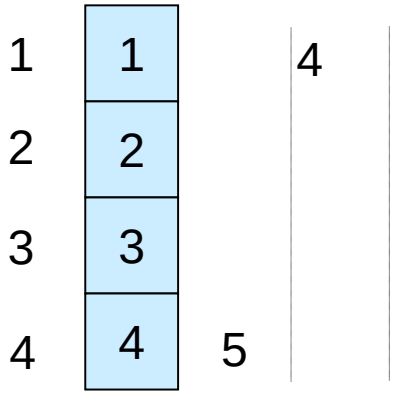


Substituição de uma página ativa → logo após a substituição, terá que ser trazida de volta

A substituição ruim (substitui justamente uma página que seria necessária) aumenta a taxa de falha de página e diminui a velocidade de execução de um processo, mas não causa execução incorreta!

Algoritmo ótimo

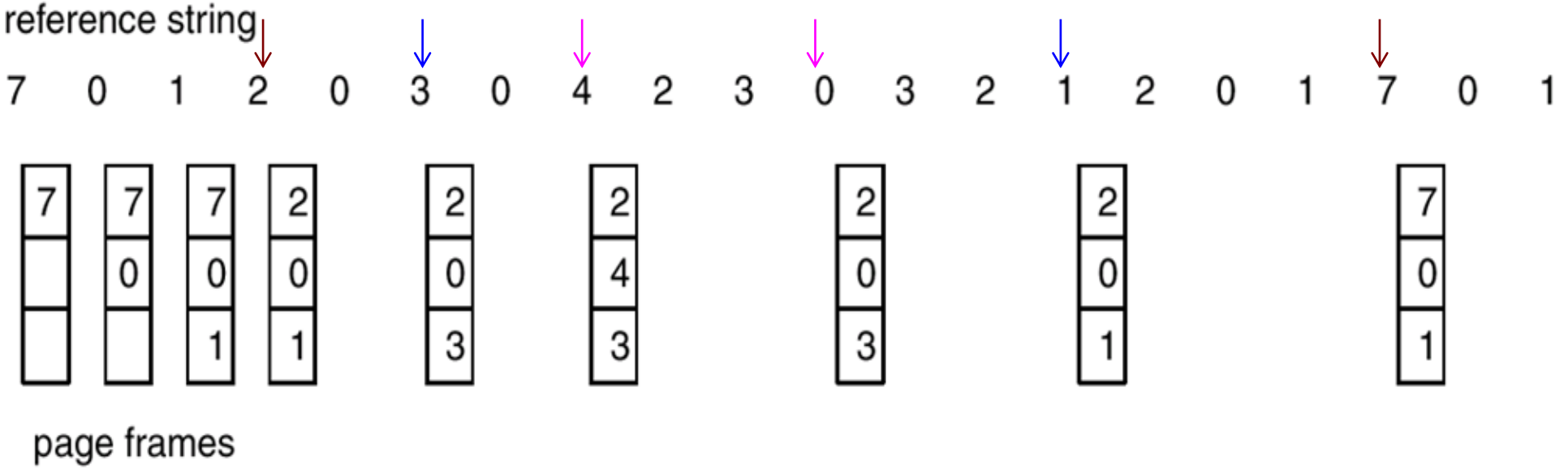
- Substituir a página que não será usada pelo período mais longo
- Exemplo com 4 quadros
- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



6 falhas de página

- Requer conhecimento futuro do string de referência
- Um algoritmo de substituição de página ótimo possui a menor taxa de falha de páginas de todos os algoritmos

Substituição ótima de página



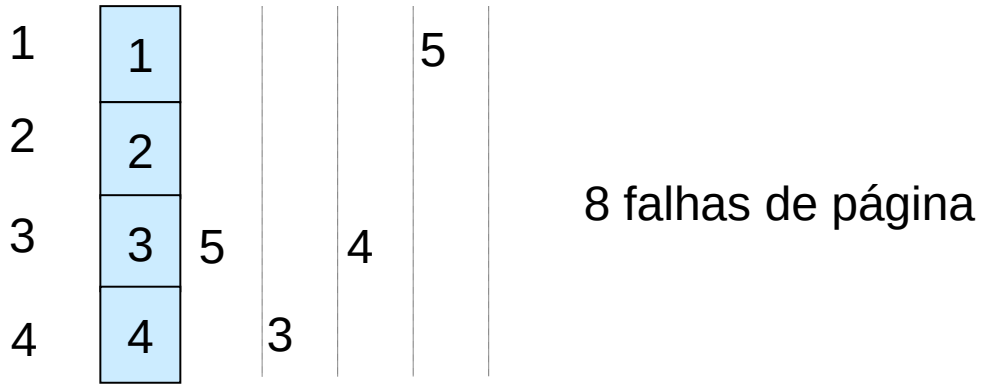
(Algoritmo Ótimo produziu 9 faltas, contra 15 do FIFO !)

- Se ignorarmos as 3 primeiras falhas (todo algoritmo tem de passar), o algoritmo ótimo é duas vezes melhor que o FIFO
- O algoritmo é ótimo pois nenhum outro algoritmo pode processar essa string com 3 quadros com menos de 9 falhas de página
- Difícil de implementar pois é necessário o conhecimento da string de referência (todas as requisições futuras)
- O algoritmo ideal é utilizado para estudos comparativos entre os algoritmos

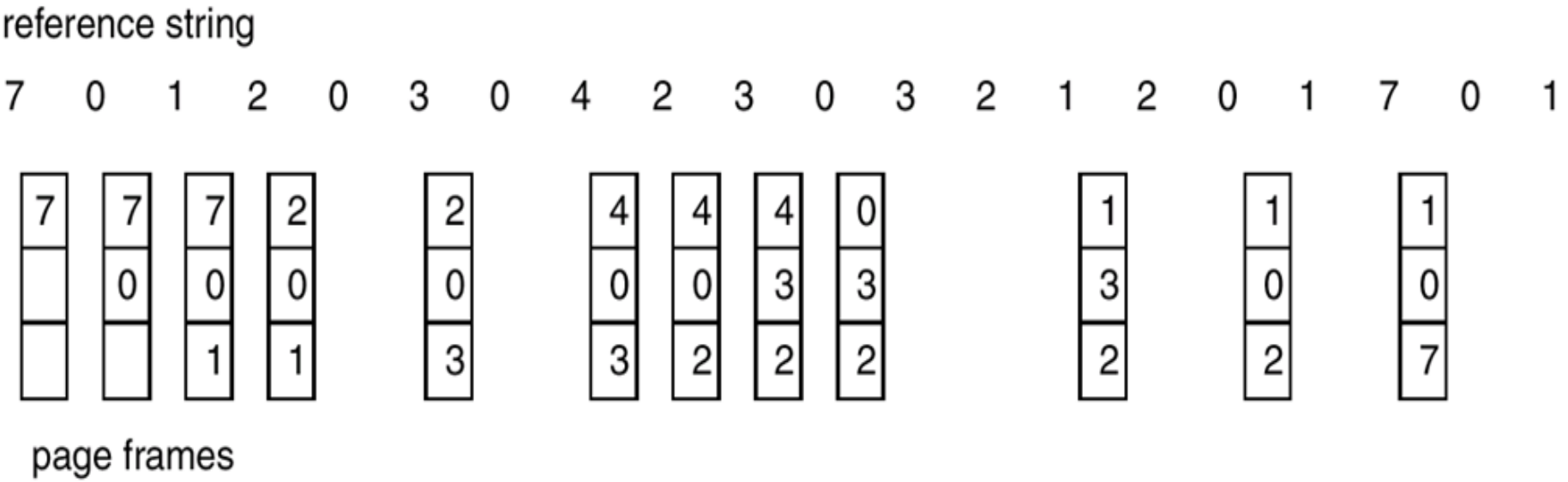
Algoritmo Least Recently Used (LRU)
(Página Menos Usada Recentemente)

- Substitui a página que não foi usada pelo maior período de tempo (passado recente)
- Uma página não usada recentemente pode não ser mais necessária (ex.: páginas de inicialização de módulo)

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



Substituição de página LRU



(O LRU produziu 12 faltas, contra 15 do FIFO e 9 do Ótimo)

Algoritmo LRU (cont.)

- **Implementação com contadores**
  - Utiliza um relógio ou contador lógico (na CPU); o relógio é incrementado a cada referência à memória
  - Toda página tem um campo tempo para indicar o “horário” da última referência na tabela de página
  - Sempre que uma página é referenciada, copiar o *clock* da CPU no campo tempo da página
    - Dessa forma, temos o “horário” da última referência à página
  - Quando uma página precisa ser trocada, buscar a página com o tempo mais antigo

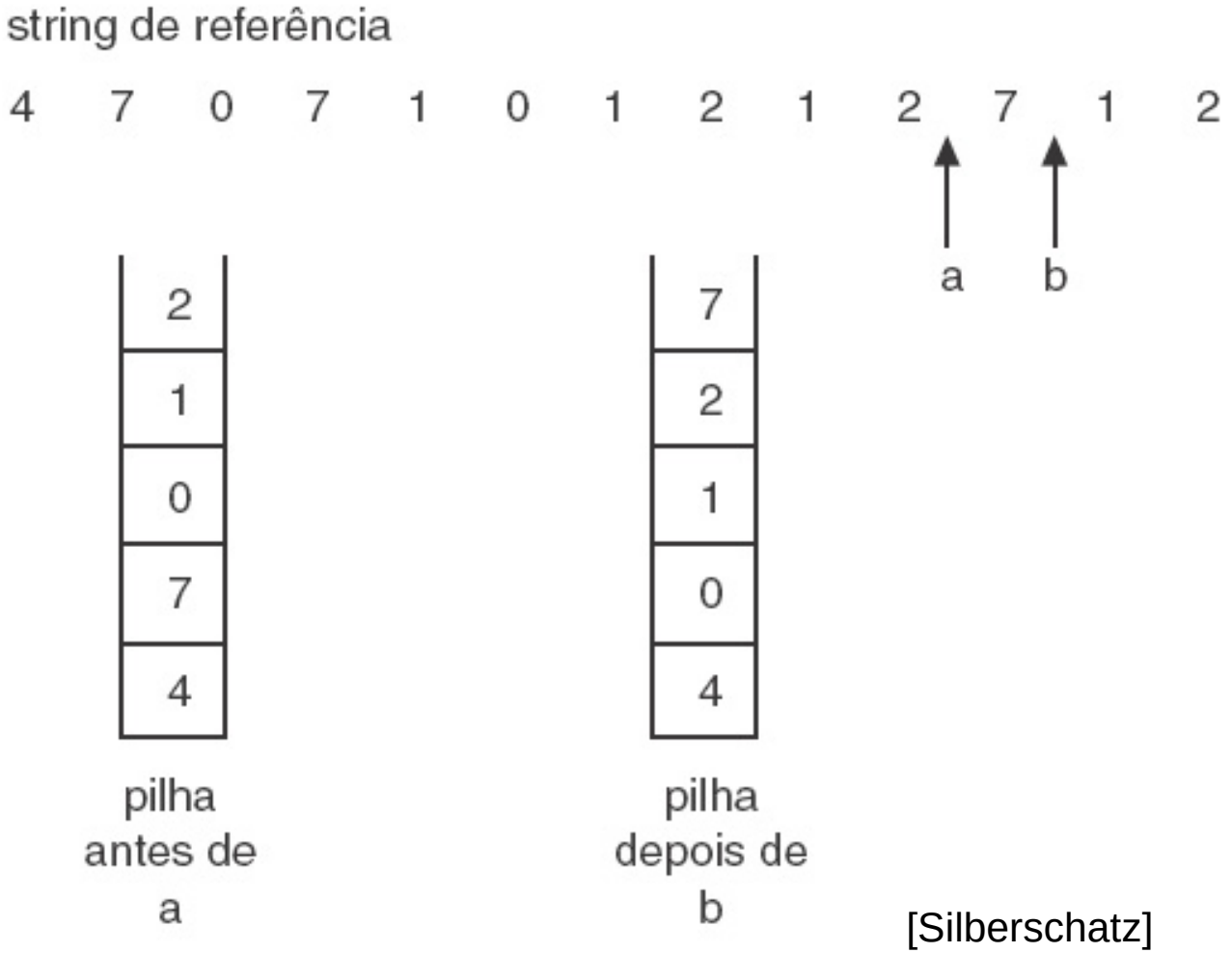
  - Requer suporte de hardware (atualizar contadores)

Algoritmo LRU (cont.)

➢**Implementação com pilha** – manter uma pilha de números de página (numa lista duplamente encadeada):

- Página referenciada:
  - ela é removida da pilha e colocada no topo
  - no topo da pilha sempre estará a página mais recentemente usada e a base da pilha é a página LRU
  - remover uma página e colocá-la no topo da pilha requer mudar 6 ponteiros no pior dos casos
- Não requer busca para substituição (topo)
- Requer suporte de hardware para atualizar a pilha

Uso de uma pilha para registrar as referências de página mais recentes



[Silberschatz]

Alocação de quadros

- Como alocar a quantidade de memória livre (quadros) entre os diversos processos?
- **Alocação igual:** Todos os processos tem o mesmo número de quadros
  - m quadros e n processos: cada processo tem m/n quadros
  - Ex.: 83 quadros e 4 processos: cada um recebe 20 quadros e sobram 3 (quadros livres)
- **Alocação proporcional:** Aloca de acordo com o tamanho do processo
  - S = tamanho do processo (número de páginas)
  - T = tamanho total dos processos (soma de S)
  - M = número total de quadros
  - A = S/T * M (número de quadros para um processo)
  - Ex.: dividir 62 quadros entre 2 processos, um com 10 páginas e outro com 127:
  - 10 / 137 * 62 = 4 (quadros para o primeiro processo)
  - 127 / 137 * 62 = 57 (quadros para o segundo processo)
- **Alocação por Prioridade:** Usa o tamanho e prioridade dos processos

Alocação de quadros

- Quando ocorre uma falha de página e não há quadro livre, é preciso obter um espaço livre, pode ser através de:

Substituição Global: a seleção do quadro a ser substituído é feita do conjunto de todos os quadros na memória, mesmo que o quadro selecionado esteja alocado a um outro processo

- Um processo pode tomar o quadro de outro processo
- É normalmente utilizado em SO
- Não é possível controlar a taxa de falha de página (fica dependente da paginação de outros processos também)

Substituição Local: a seleção do quadro é feita a partir do próprio conjunto de quadros alocados ao processo

- A quantidade de quadros alocados para o processo não muda
- Não interfere na execução de outros processos
- Pode aumentar a taxa de falha de página (quando precisar mais quadros e por não poder alocar quadros a mais do que já tem)

Thrashing

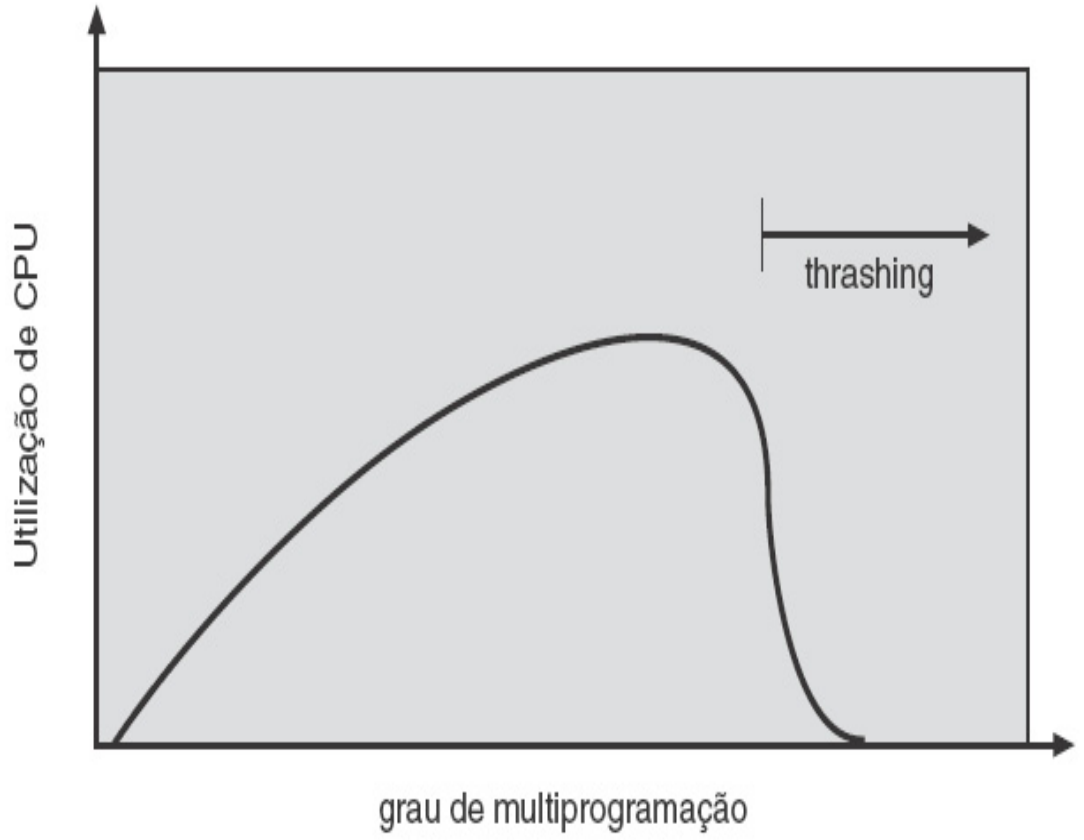
➢Se um processo não tem quadros “suficientes”, a **taxa de falha de página** é **muito alta** (requer a frequente substituição de uma página por outra para executar). Disso resulta:

- baixa utilização da CPU (processos suspensos por paginação)
- o sistema operacional “pensa” ser necessário aumentar o grau de multiprogramação
- outro processo é acrescentado ao sistema
- a taxa de falha de página torna-se ainda mais alta

➢***Thrashing*** ≡ quando um processo gasta mais tempo paginando do que executando

- Resulta em graves problemas de desempenho

Thrashing (cont.)



[Silberschatz]

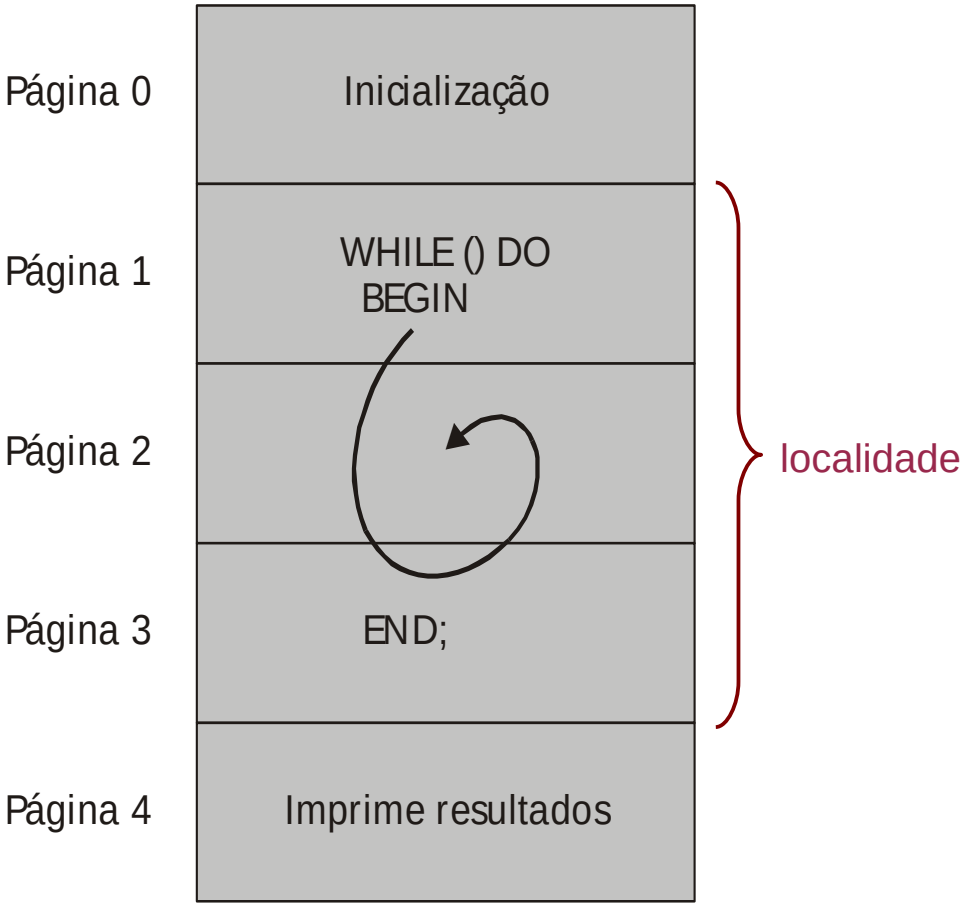
Modelo de localidade

➢Para previnir o *thrashing*, deve-se fornecer a cada processo o número de quadros necessários

➢Mas como saber quantos quadros um processo vai precisar?

- Uma estratégia é utilizar o modelo da localidade de execução:
  - Uma localidade é um conjunto de páginas que são usadas ativamente juntas
  - Por exemplo, quando uma função é chamada, essa chamada define uma localidade (as referências à memória são feitas com relação às instruções da função, às variáveis locais, etc.)
  - Quando a execução da função termina, o processo deixa a localidade (não utiliza mais as variáveis locais e instruções da função)
  -

Modelo de localidade

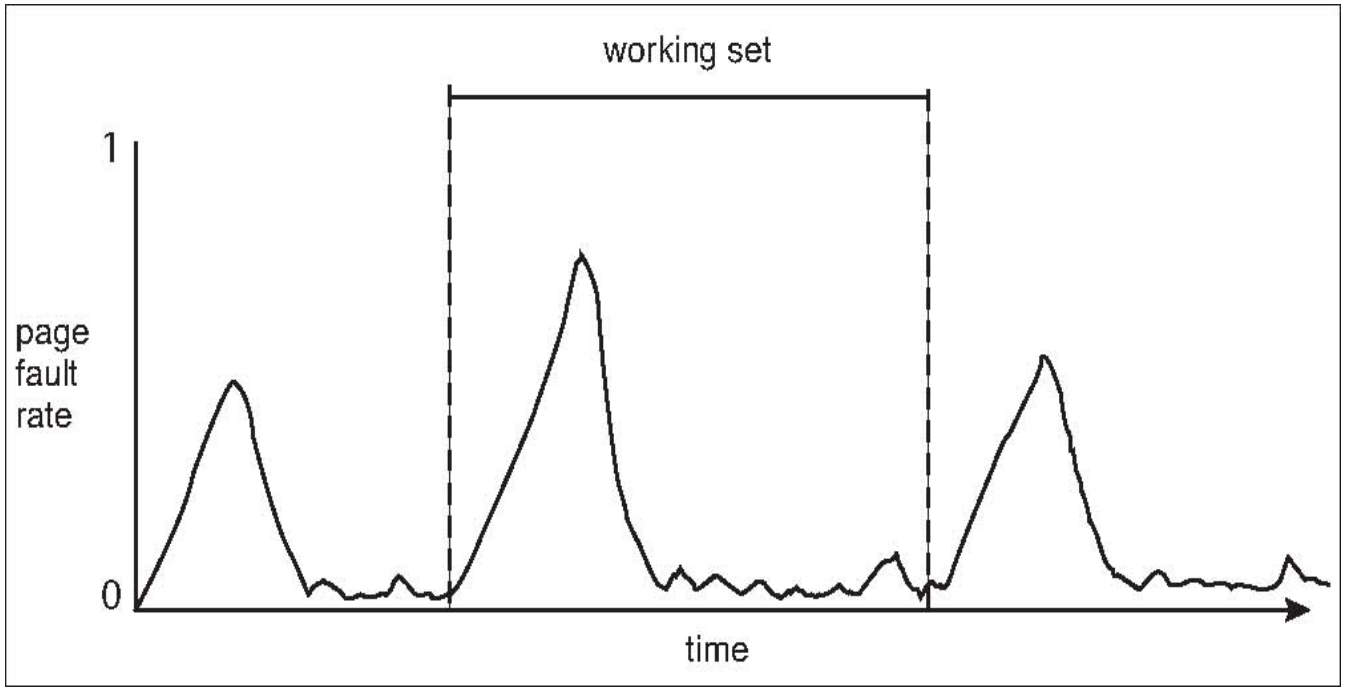


[Machado]

➢O princípio de localidade na prática significa que o processador tende a concentrar suas referências a um conjunto de páginas do processo durante um tempo

➢Na figura, no laço while (3 páginas), a probabilidade de que essas 3 páginas sejam referenciadas várias vezes é bem alta

Falha de página e conjunto de trabalho

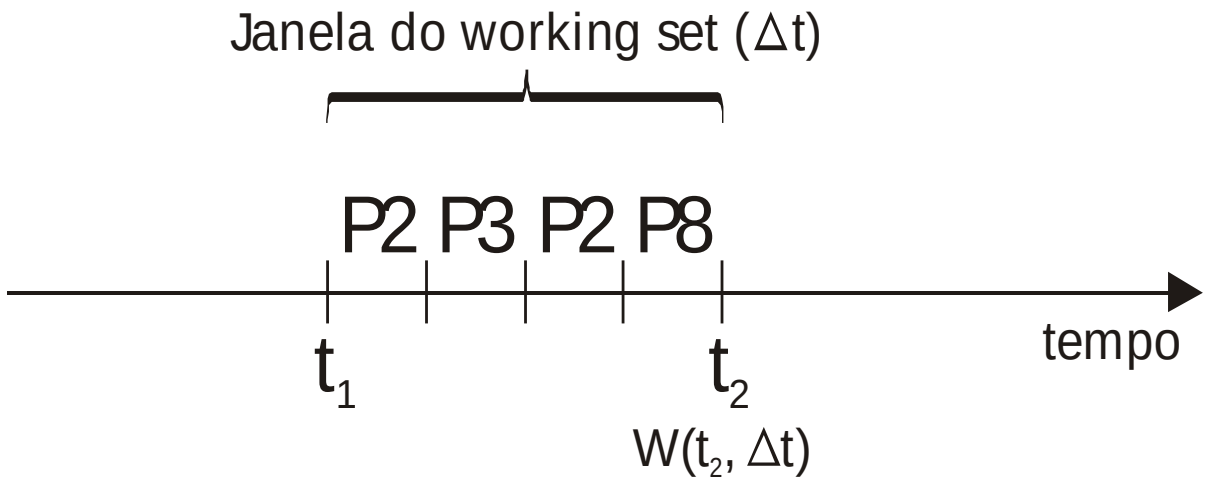


[Silberschatz]

- Em geral, no início da execução há um grande número de falha de página, mas com o tempo a tendência é que esse número decresça e se estabilize por um tempo
- Após um determinado tempo, o processo gera novamente uma elevada taxa de paginação e isso se repete inúmeras vezes durante a execução do processo
- Localidade: as referências aos endereços de um processo concentram-se em um número determinado de páginas

Modelo de conjunto de trabalho (working-set)

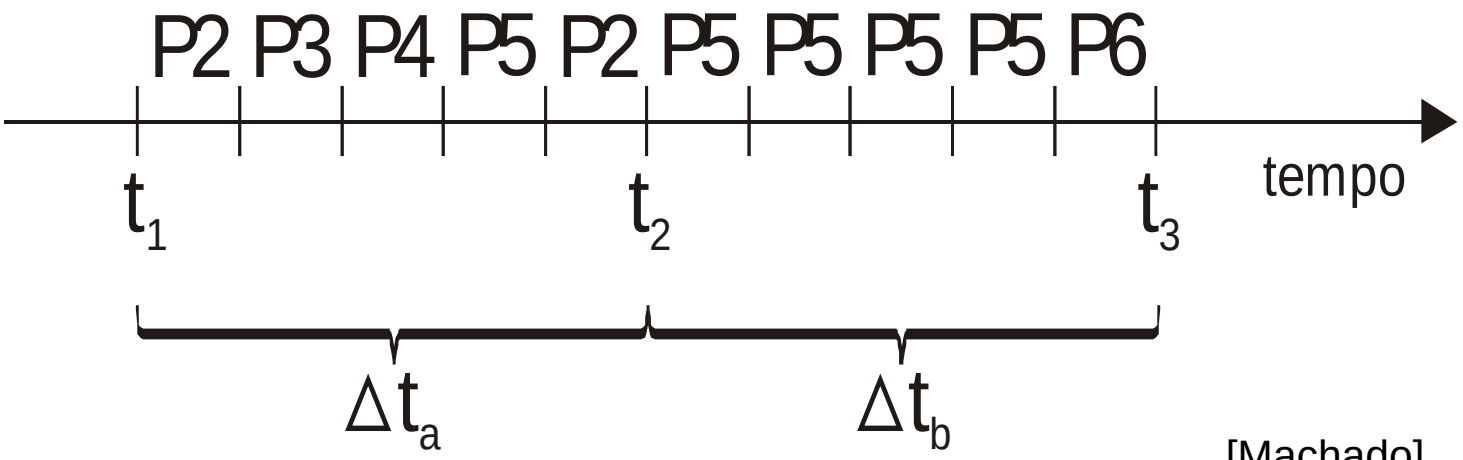
- Como saber o tamanho de uma localidade?
- *Working set* é definido como um conjunto de páginas referenciadas por um processo durante um intervalo de tempo



[Machado]

Modelo de conjunto de trabalho (working-set)

- Como saber o tamanho de uma localidade?
- *Working set* é definido como um conjunto de páginas referenciadas por um processo durante um intervalo de tempo



[Machado]

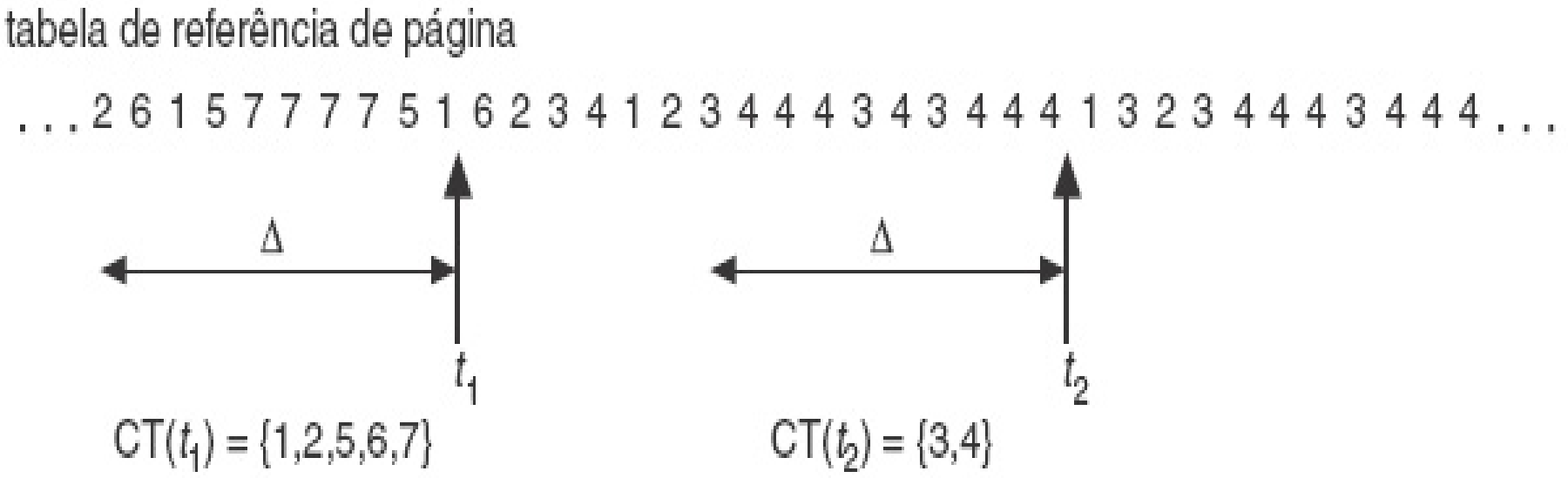
O número de páginas distintas referenciadas é o tamanho do *working set*
Isso permite prever quantas páginas são necessárias para execução
Localidade varia ao longo da execução e também o *working set*

Modelo de conjunto de trabalho (working-set)

- $\Delta \equiv$ janela de conjunto de trabalho $\equiv$ um número fixo de referências de páginas
- WSS_i (conjunto de trabalho do Processo P_i) = número total de páginas referenciadas na mais recente janela Δ (varia no tempo)
  - se Δ r muito pequeno, ele não abrangerá a localidade toda
  - se Δ r muito grande, ele poderá se sobrepor a várias localidades
  - Se Δ ∞ ele vai abranger todo o programa
- $D = \sum WSS_i \equiv$ Demanda total por quadros
- se $D > m \rightarrow$ *Thrashing* (Demanda maior que o total de quadros disponíveis)
  - Estratégia: se $D >$ m, suspender um dos processos

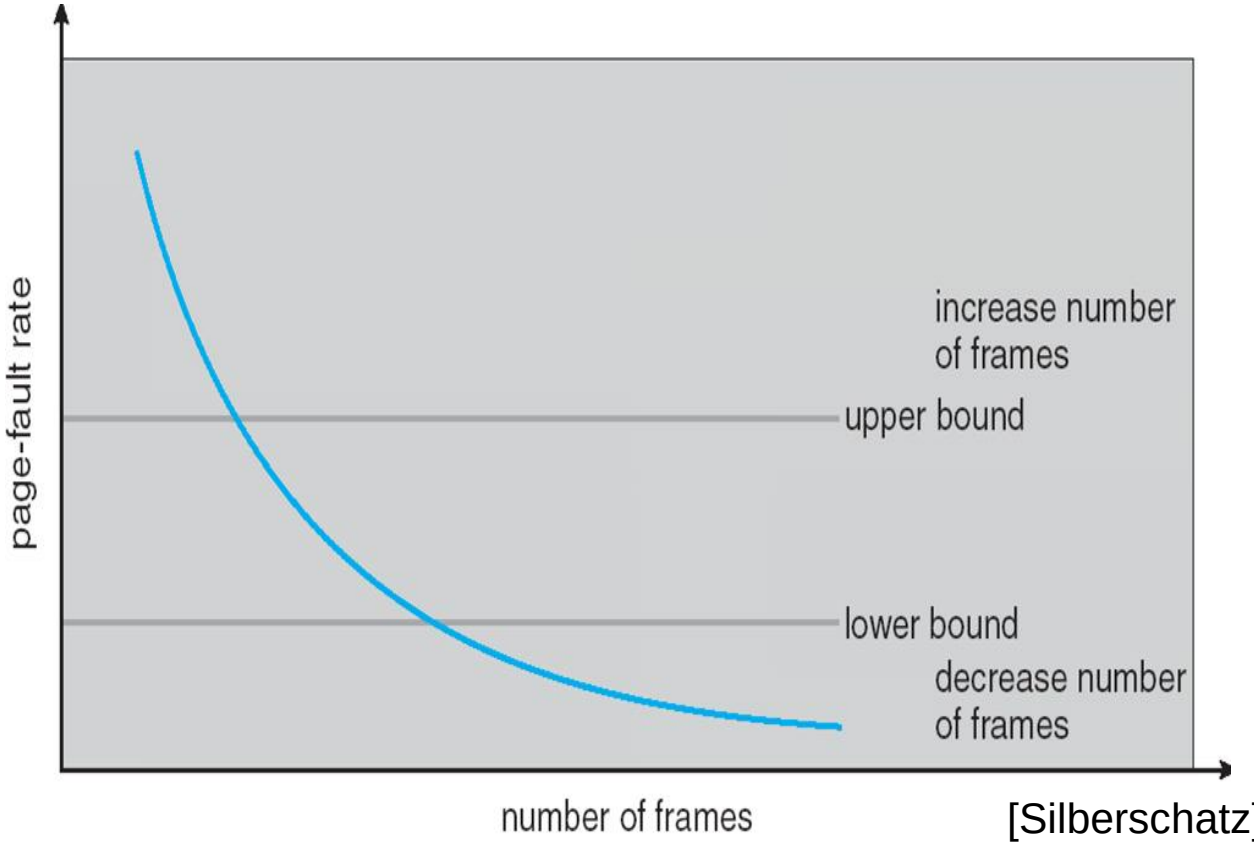
Modelo de working-set (cont.)

- Considere Δ = 10



[Silberschatz]

Frequência de falha de página



[Silberschatz]

- Quando a taxa de falha de página é muito alta, o processo precisa de mais quadros, e ao contrário, se for muito baixa, os quadros estão sobrando
- Pode-se estabelecer limites superior e inferior para a taxa de falha de página (se ultrapassar o limite máximo, deve-se alocar quadro, e se ficar abaixo do limite inferior, remove-se quadro)
- Controle da taxa de falha de página, evita-se o *trashing*

Referências

- [Silberschatz] SILBERCHATZ, A., GALVIN, P. B. e GAGNE, G. **Sistemas Operacionais com Java**. 7ª ed., Rio de Janeiro: Elsevier, 2008.
- [Tanenbaum] TANENBAUM, A. **Sistemas Operacionais Modernos**. 3ª ed. São Paulo: Prentice Hall, 2009.
- [MACHADO] MACHADO, F. B. e MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 4ª ed., Rio de Janeiro: LTC, 2007.