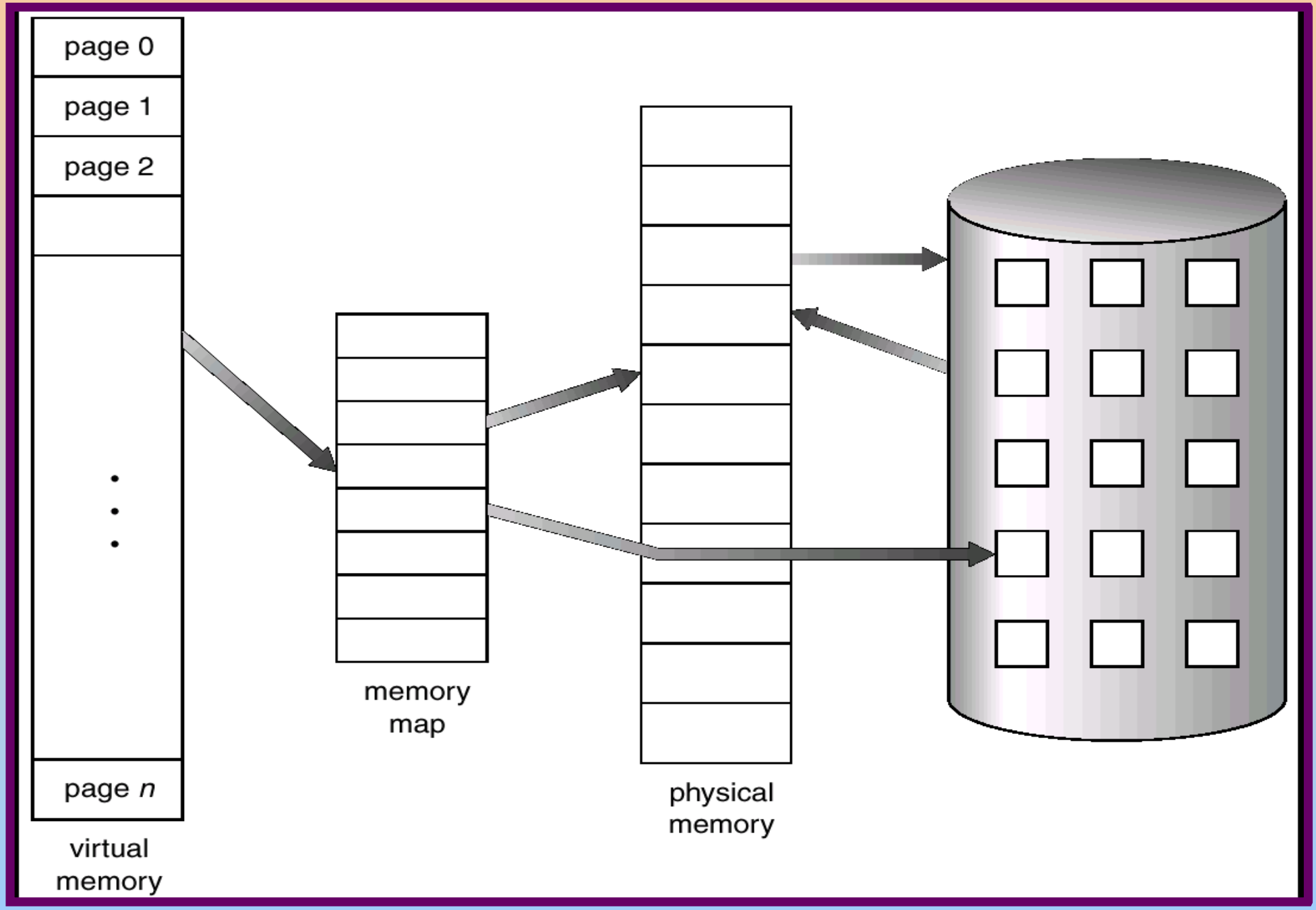
Memória Virtual
parte1

- Conceitos básicos
- Paginação por demanda
- Criação de processos
- Substituição de páginas
- Exemplos em sistemas operacionais

Conceitos básicos

- **Memória virtual** – separação de memória lógica da memória física.
  - Somente parte do programa precisa estar na memória para execução.
  - Espaço de endereçamento lógico pode ser muito maior que o espaço de endereçamento físico.
  - Permite que espaços de endereçamento sejam compartilhados por vários processos.
  - Permite uma criação mais eficiente de processos.
- Memória virtual pode ser criada via:
  - Paginação por demanda (Demand paging)
  - Segmentação por demanda.

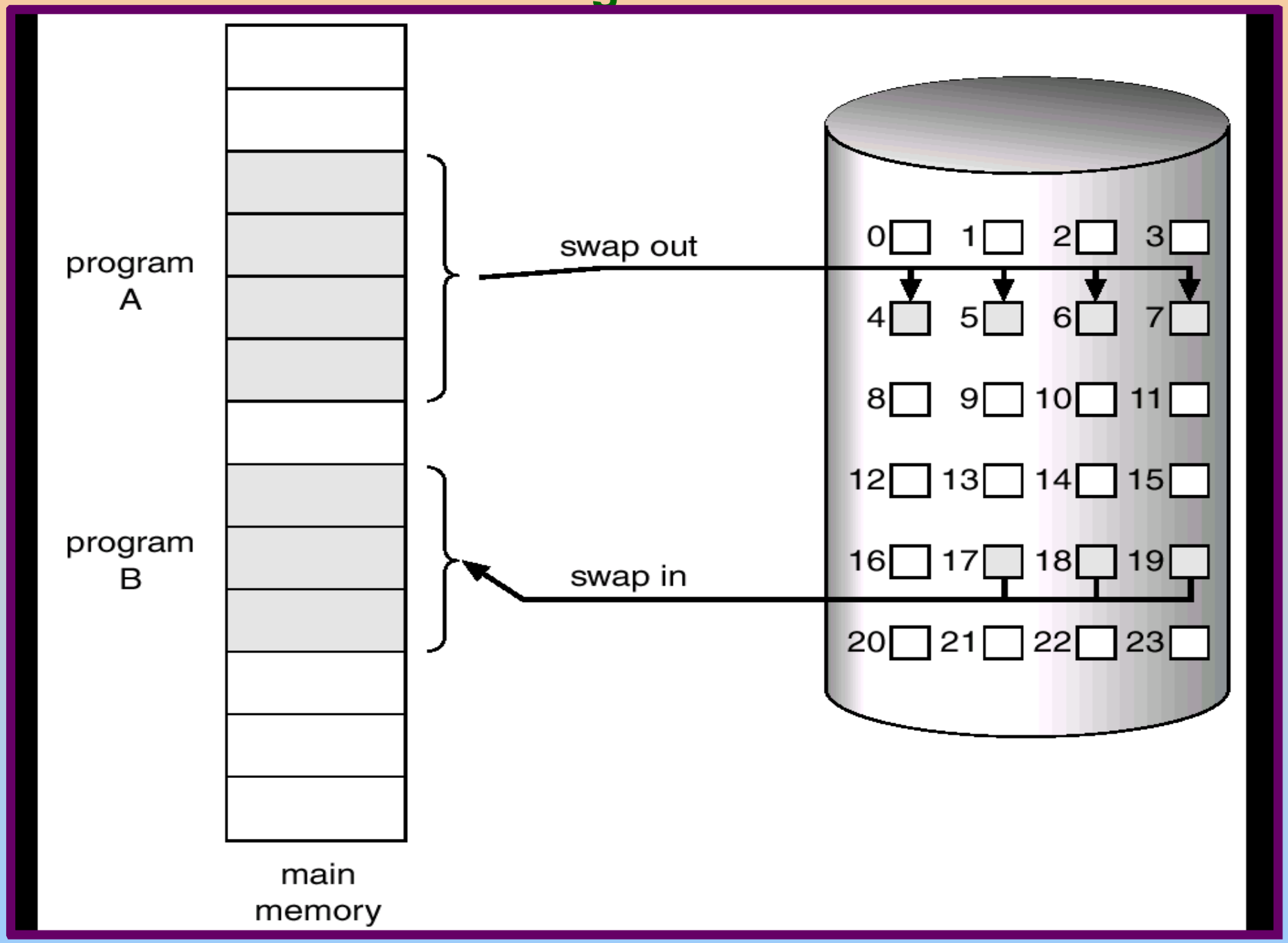
Memória virtual que é maior que a memória física



Paginação por demanda

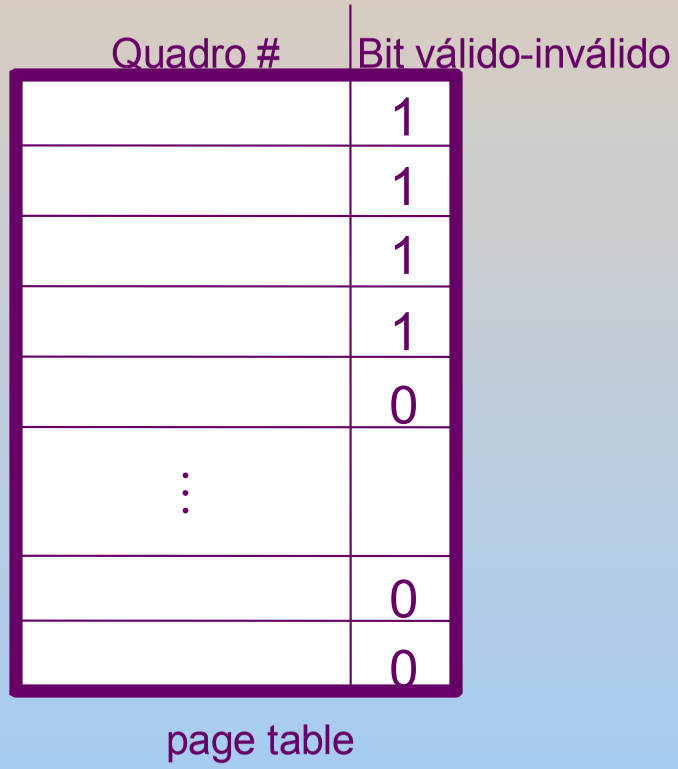
- Coloca uma página na memória somente quando é necessário.
  - Necessita de menos I/O
  - Necessita de menos memória
  - Resposta mais rápida
  - Mais usuários
- Se uma página é necessária $\Rightarrow$ basta referenciá-la
  - Referência inválida $\Rightarrow$ erro
  - Não está na memória $\Rightarrow$ carregá-la na memória

Transferência de uma memória paginada para espaço contíguo no disco



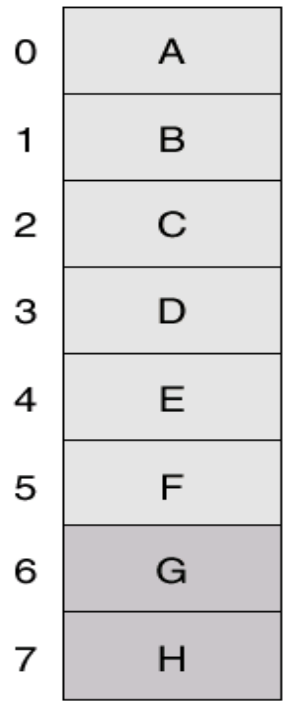
Bit Válido-Inválido

- Com cada entrada na tabela de página uma bit válido-inválido é associado(1 ⇒ na memória, 0 ⇒ não está na memória)
- Inicialemnte, todos os bits estão em 0.
- Exemplo:

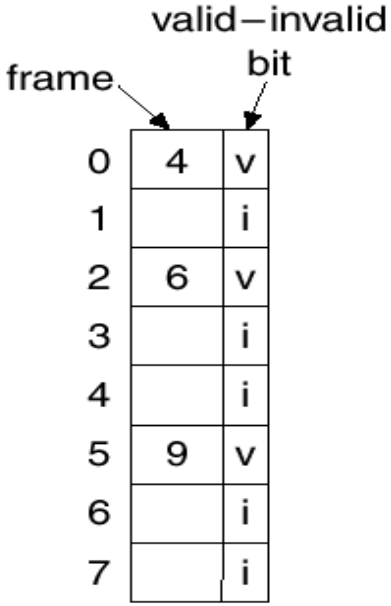


- Durante a tradução de nedereço, se o bit válido-inválido na

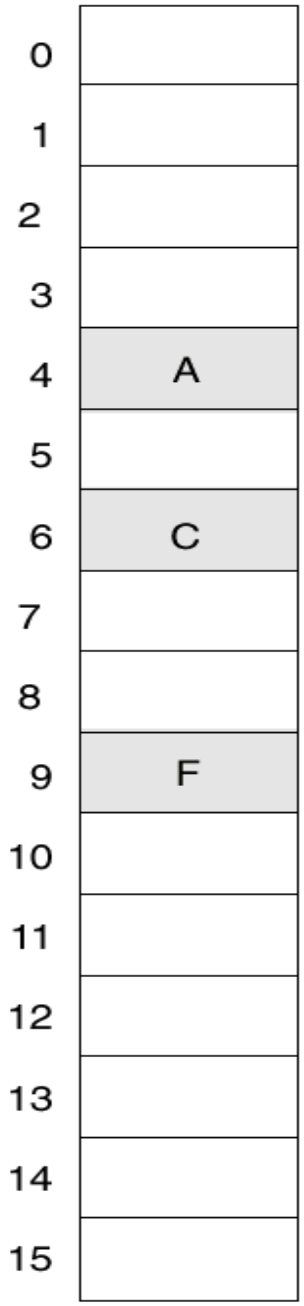
Tabela de páginas com algumas páginas que não estão na memória



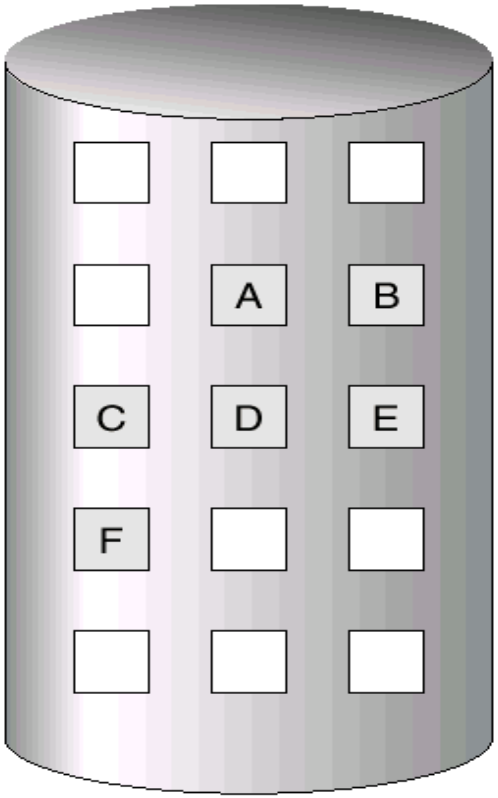
logical memory



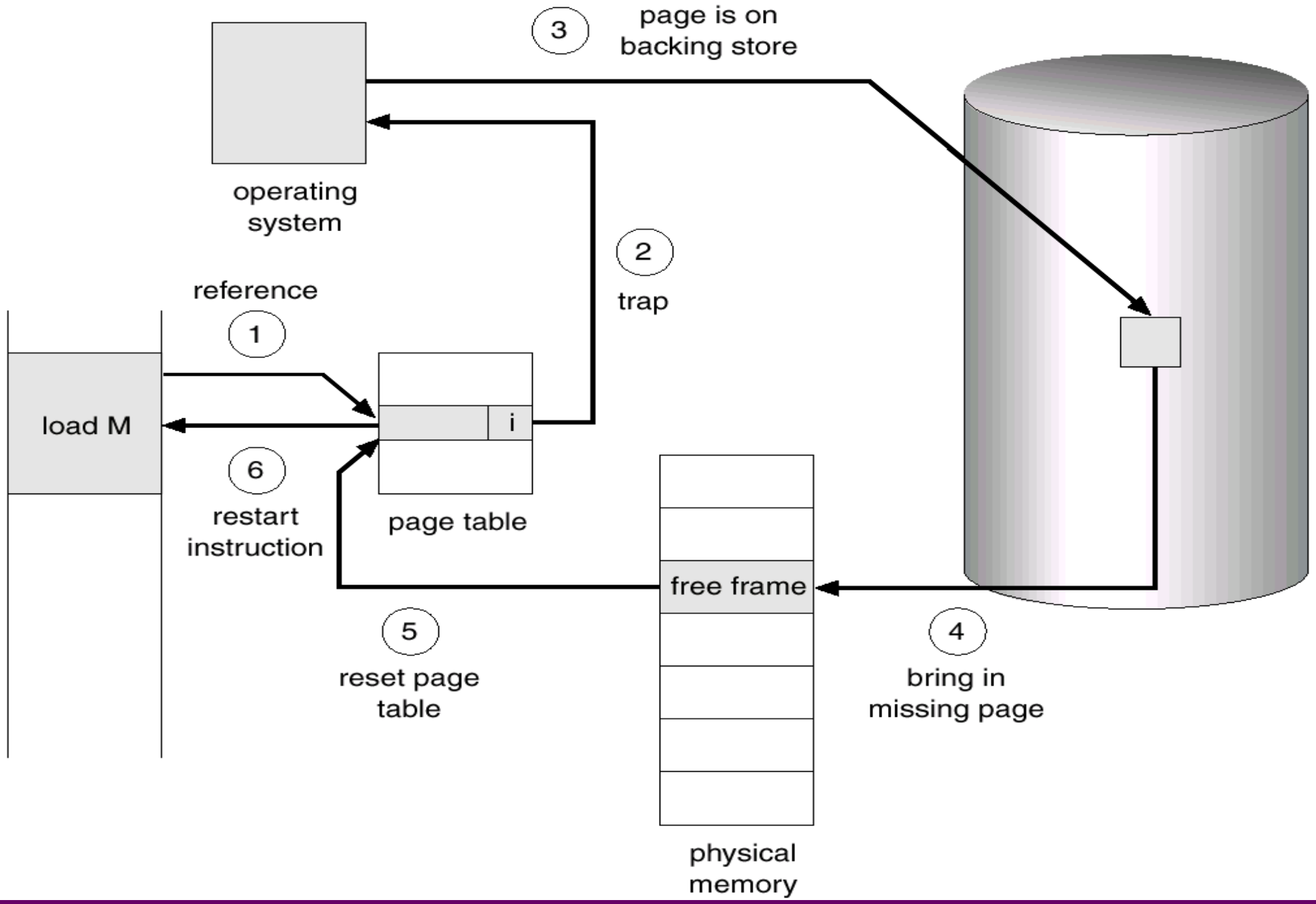
page table



physical memory



Passos no tratamento de falha de



O que acontece se não existem quadros livres?

- Substituição de páginas – encontrar alguma página em memória e armazená-la em disco.
  - algoritmo
  - performance – queremos um algoritmo que resulte no menor número de falhas de página.
- Algumas páginas podem sair da memória várias vezes.

Criação de processos

- Memória virtual permite outros benefícios durante a criação de processos:

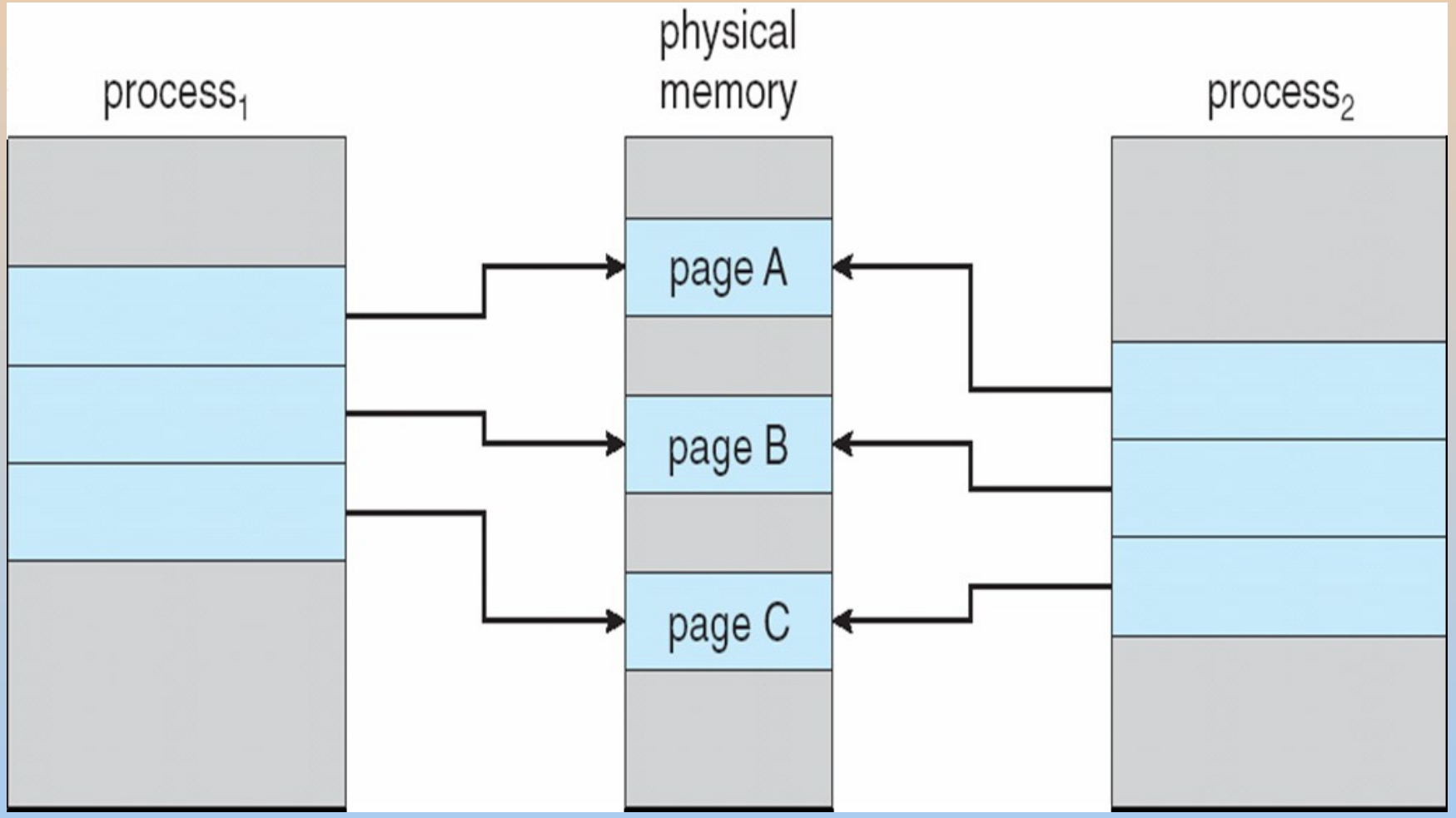
  - Cópia-na-escrita(Copy-on-Write)

  - Arquivos mapeados em memória(Memory-Mapped Files)

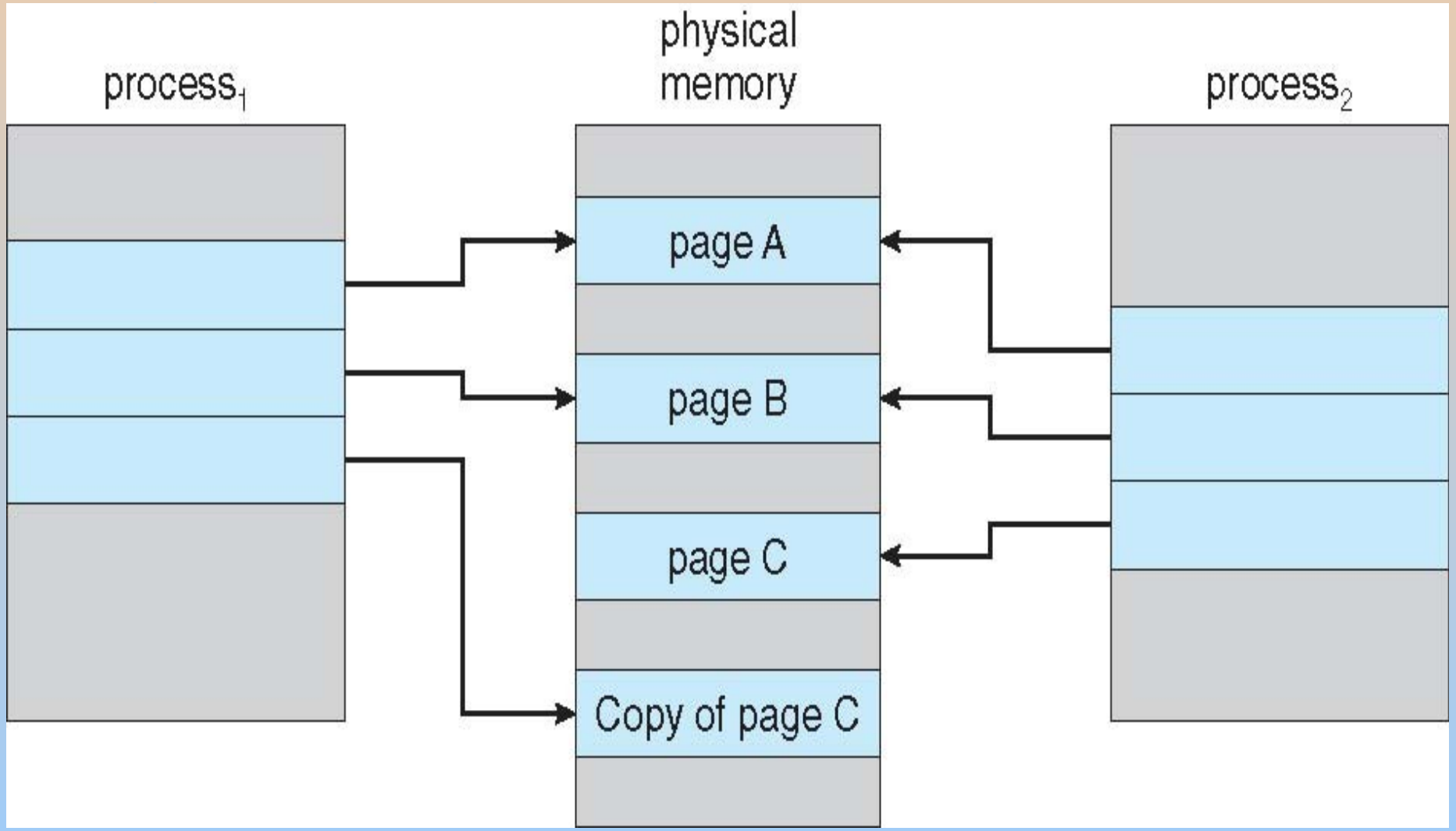
Copy-on-Write

- Copy-on-Write (COW) permite que pai e filho inicialmente compartilhem as mesmas páginas em memória.
- Se um dos processos modifica a página compartilhada, somente a página é copiada.
- COW permite criação mais eficiente de processos, uma vez que somente páginas modificadas sejam copiadas.

COW antes do Process 1 modificar a pagina C



COW: Depois do Process 1 modificar a pagina C



Arquivos mapeados em memória

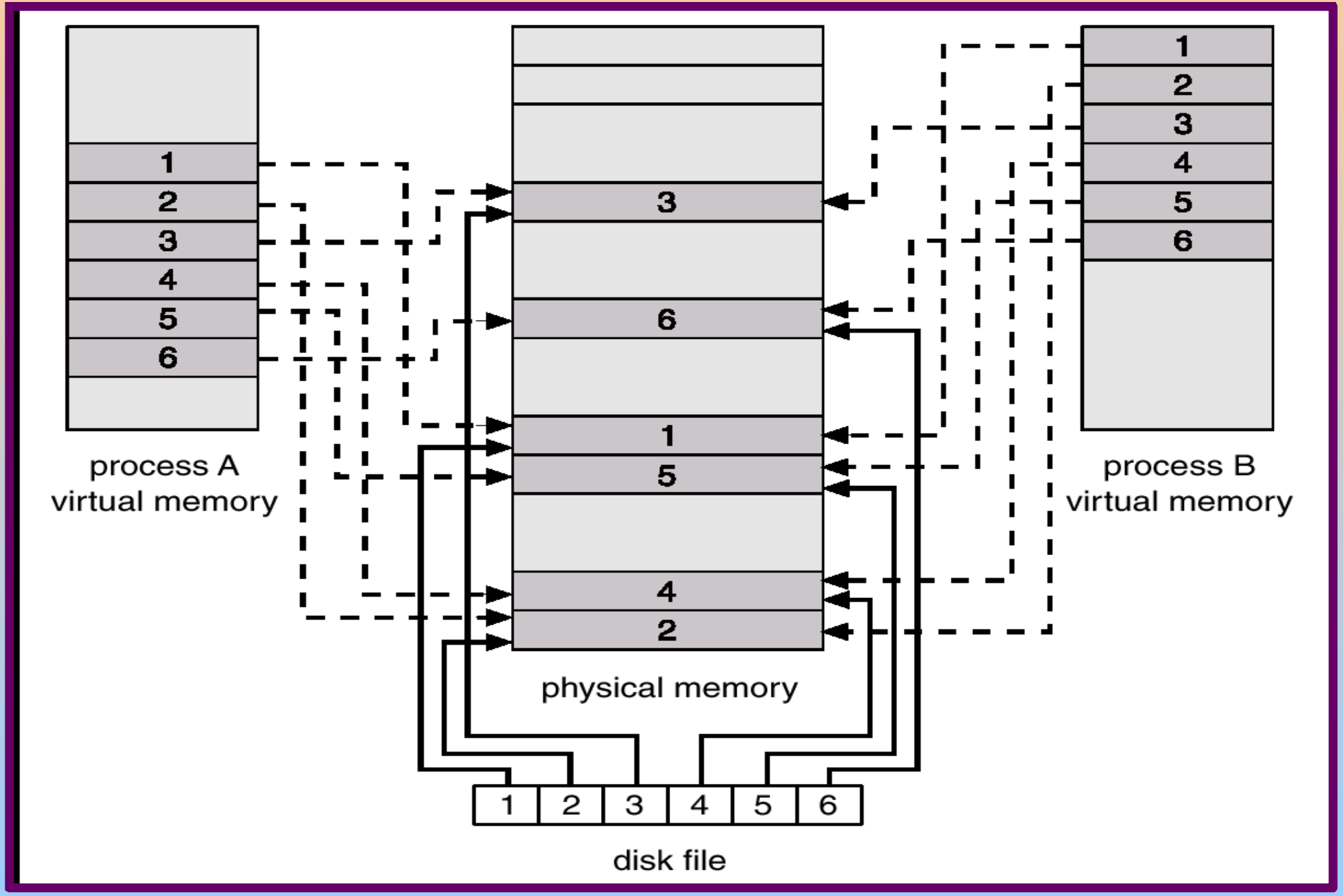
- I/O de arquivos mapeados em memória permitem que I/O em arquivos seja tratada como uma rotina de acesso à memória por mapeamento de um bloco de disco numa página de memória.

- Um arquivo é inicialmente lido usando paginação por demanda. Um trecho do arquivo, do tamanho de uma página, é lido do sistema de arquivos para a memória física. Escritas/leituras subsequentes são tratados como acessos ordinários a memória.

- Simplifica o acesso a arquivos tratando I/O de arquivos através de acessos à memória ao invés de systems calls do tipo **read()** e **write()** .

- Permite, também, que vários processos possam mapear o mesmo arquivo, possibilitando compartilhamento de páginas em memória.

Arquivos mapeados em memória



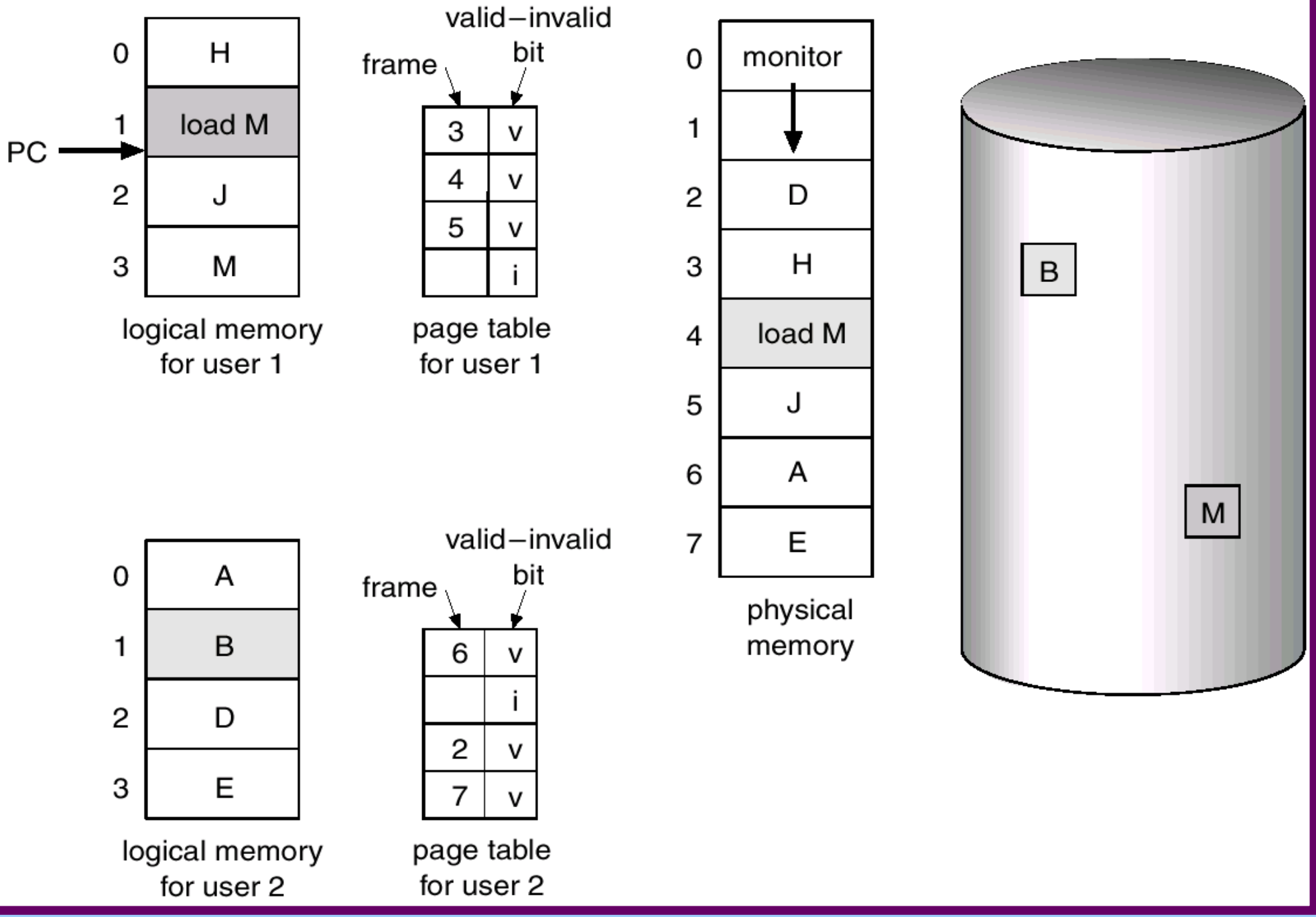
Substituição de páginas

- Previne super-alocação de memória modificando a rotina de serviço de falha de página para incluir substituição de página.

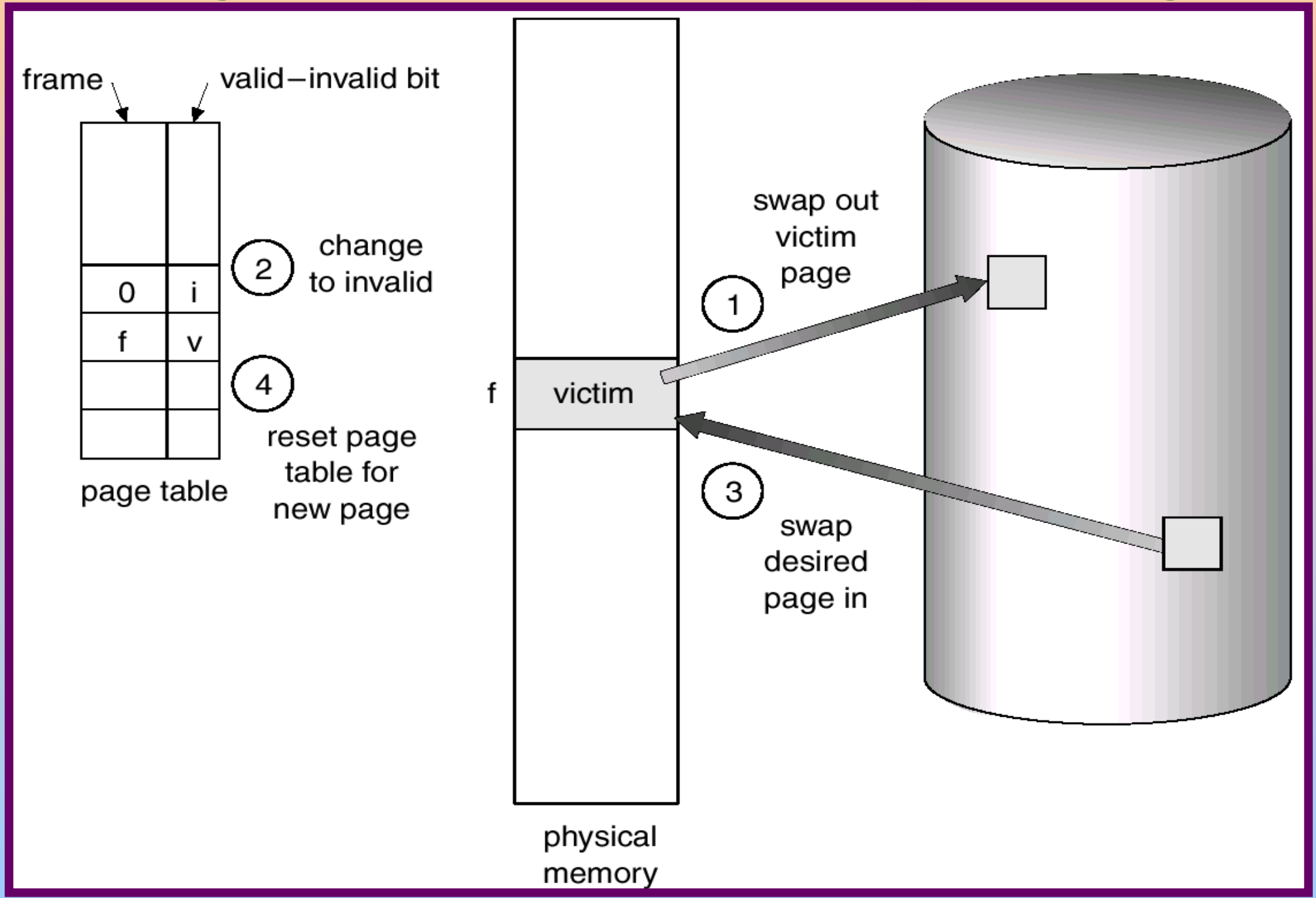
- Usa bit de modificação(dirty bit) para reduzir overhead de transferência de página – somente páginas modificadas são escritas no disco.

- Substituição de páginas completa separação entre memória lógica e memória física. Assim, uma grande memória virtual pode ser mapeada para uma pequena memória física.

Necessidade de substituição de páginas



Algoritmo básico de substituição de páginas



Algoritmos de Substituicao de Paginas

- Objetivo: queremos obter uma menor taxa de page-faults

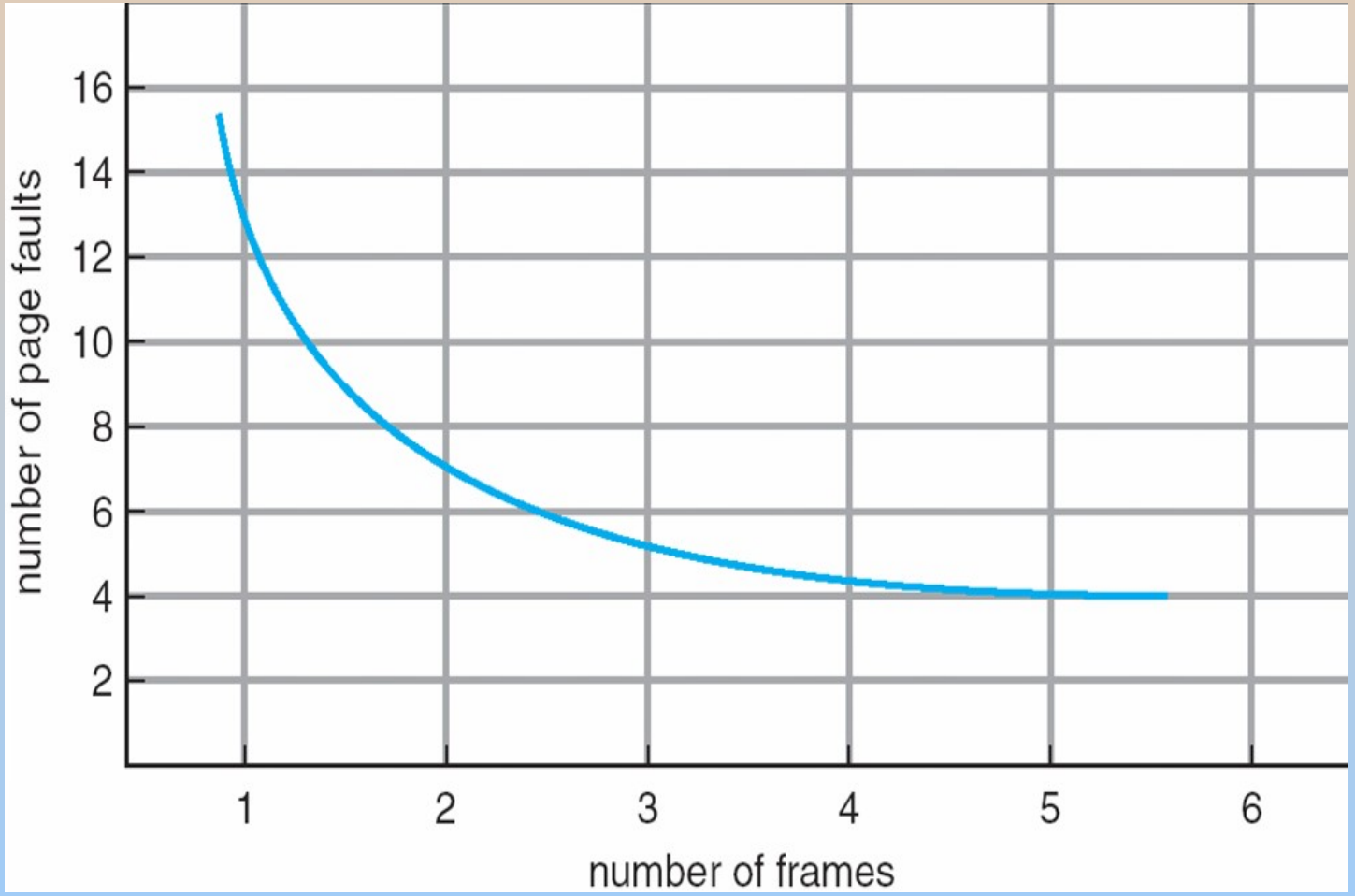
- Avaliacao dos algoritmos:
  - rodando o algoritmo com sequencias de referencias memoria
    - ✔ (seq. de enderecos referenciados ==> sequencia de paginas referenciadas ==> string de referencia
    - ✔ computar o numero de page faults para aquela sequencia
    - ✔ strings de referencia podem ser obtidos via coleta em execucoes reais:
      - simulador da arquitetural ==> rodar programa no simulador ==> salvar enderecos referenciados

- Nos nossos exemplos usaremos o seguinte string de referencia:

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

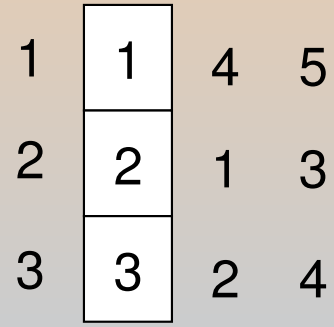
Grafico de Page Faults Versus Numero de Quadros

- Idealmente: alocando mais quadros gostariamos de obter menor quantidade de page faults (PF)
- no limite: teremos apenas os PF compulsorios



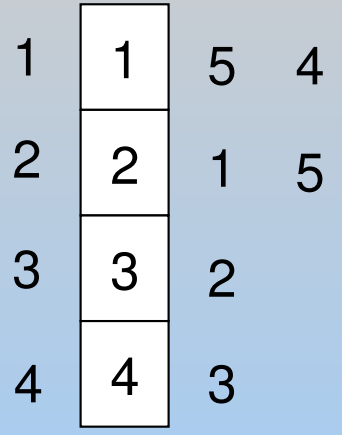
Algoritmo First-In-First-Out (FIFO)

- String de Referencia: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 quadros alocados ao processo



9 page faults

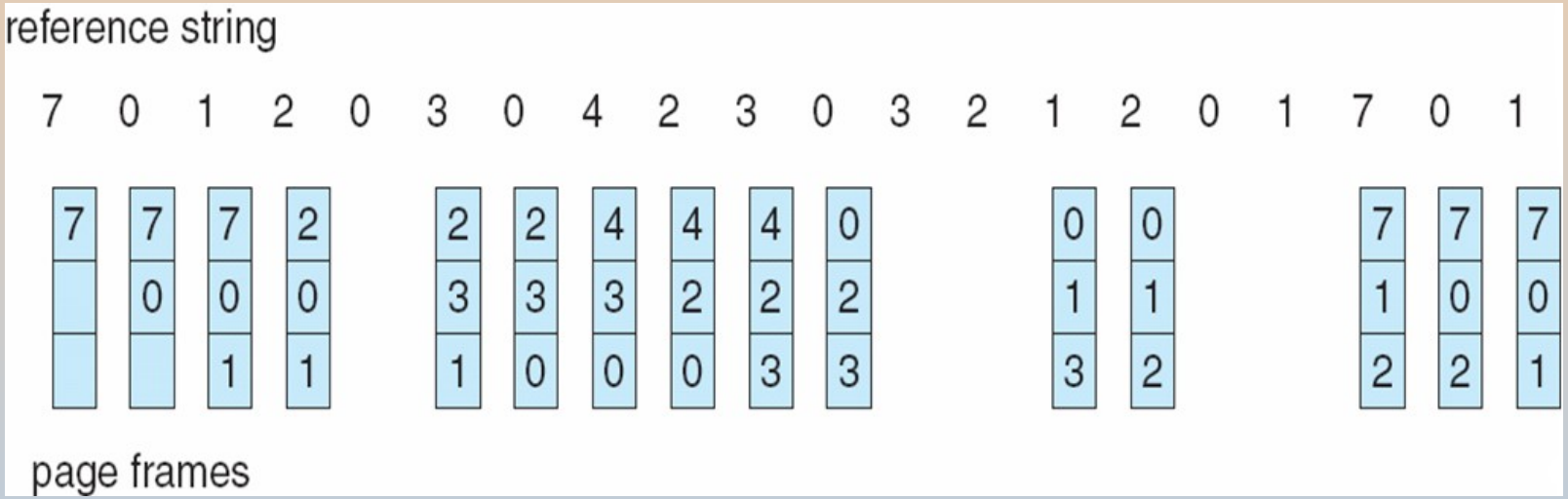
- com 4 quadros teriamos:



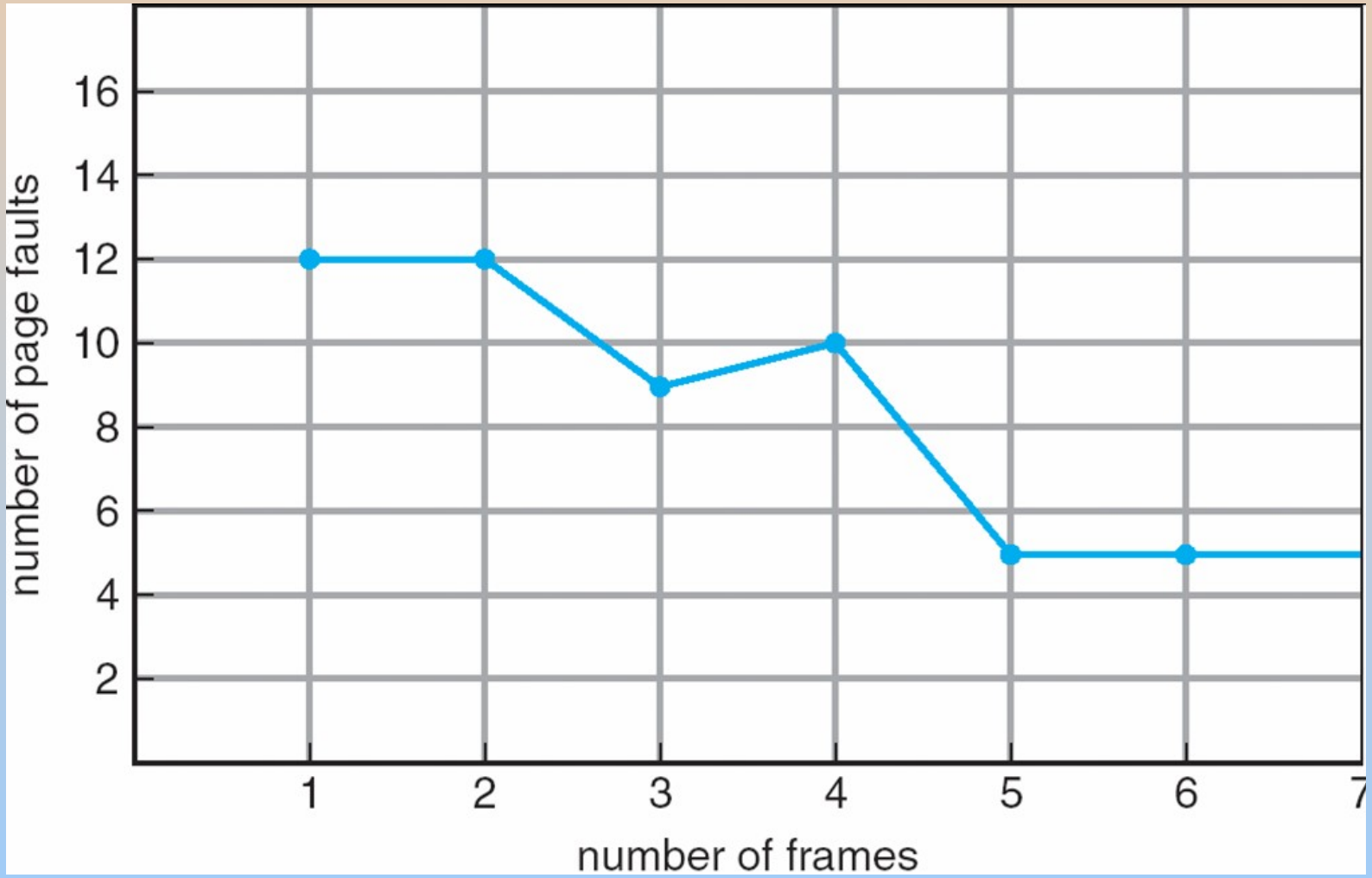
10 page faults

- FIFO sofre da Anomalia de Belady: alocando mais quadros ⇒ podemos obter mais page faults

Outro exemplo de FIFO Page Replacement



Ilustrando a anomalia de Belady para a sequencia de refs. anterior e diversas quantidades totais de quadros



FIM da parte 1

FIM da parte 1