Memória Virtual

Referência:

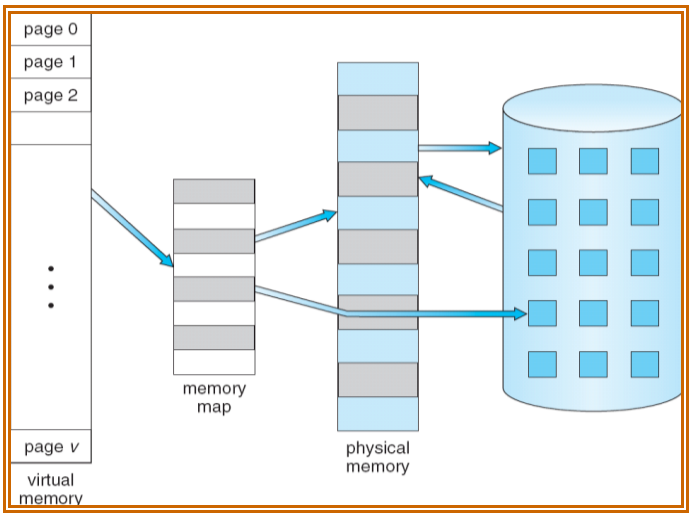
Silberschatz, Abraham. Sistemas Operacionais com Java. 7 ed. Rio de Janeiro: Elsevier, 2008.

Fundamentos

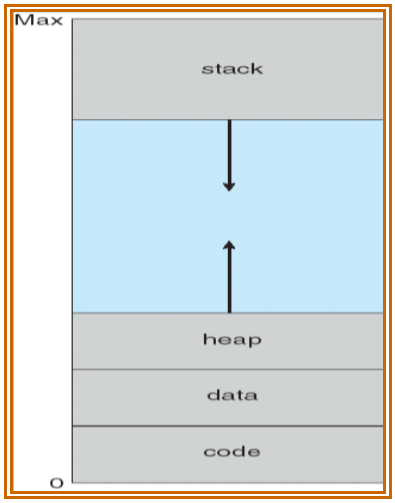
- Memória virtual - separação da memória lógica de usuário da memória física.
    - Princípio: Apenas parte do programa precisa estar na memória para a execução.
    - O espaço de endereços lógicos pode ser muito maior do que o espaço físico de endereços. (Espaço = capacidade de endereçamento = número de bits do endereço)
    - Precisa permitir que as páginas sejam *suapadas* para/do disco.
- A memória virtual pode ser implementada por:
    - Paginação por demanda.
    - Segmentação por demanda.

Na verdade o termo *swap* para páginas não está adequado, o *swap* se refere à troca de um processo na memória memória principal para a memória secundária por decisão de um escalonador de longo prazo. No caso das páginas, quem executa as trocas de páginas é o *paginador* e não necessariamente algum processo é retirado da memória principal para que uma página possa ser trazida do disco.

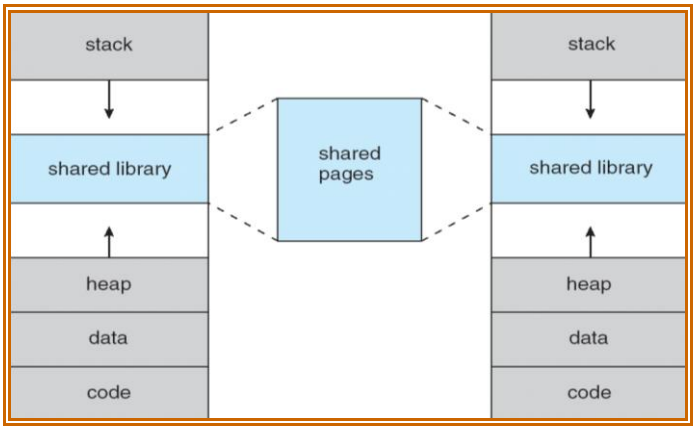
Memória virtual maior que a memória física



Espaço de endereço virtual

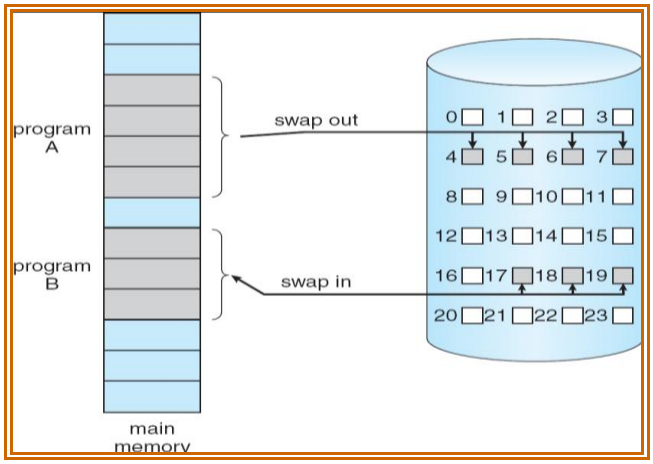


Biblioteca compartilhada usando memória virtual



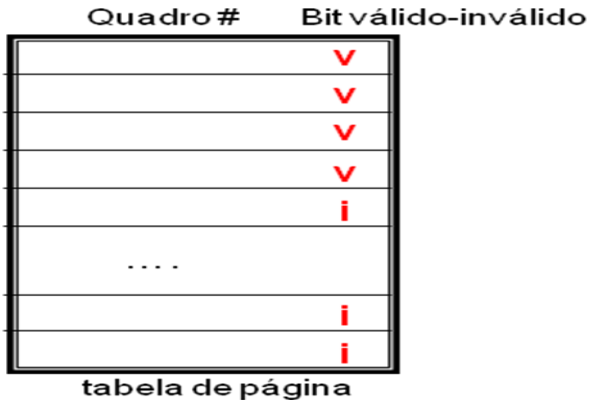
Paginação por demanda

- Traz uma página para a memória apenas se ela for necessária.
  - Minimiza as operações de E/S para trazer/guardar páginas
  - Usa o mínimo de memória
  - Resposta rápida
  - Maior quantidade de usuários
- A página é necessária => houve uma referência para ela
  - referência inválida => aborta a operação e o processo
  - não está na memória => trazer para a memória



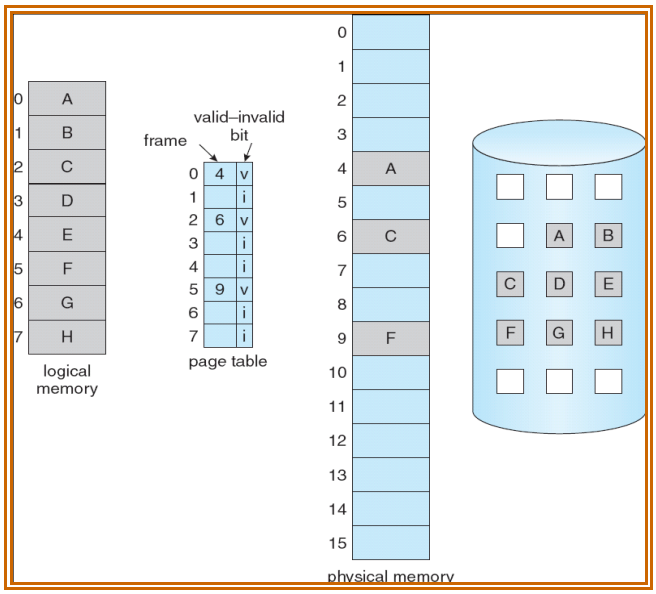
Bit de válido-inválido

- Para cada linha da tabela de páginas há um bit de válido-inválido associado (1 => está na memória, 0 => não está na memória)
- Inicialmente, todos os bits válido-inválido da tabela são colocados em 0
- Exemplo de um instante numa tabela de páginas



- Durante a tradução da página, se o bit de válido-inválido estiver em 0 => falha de página

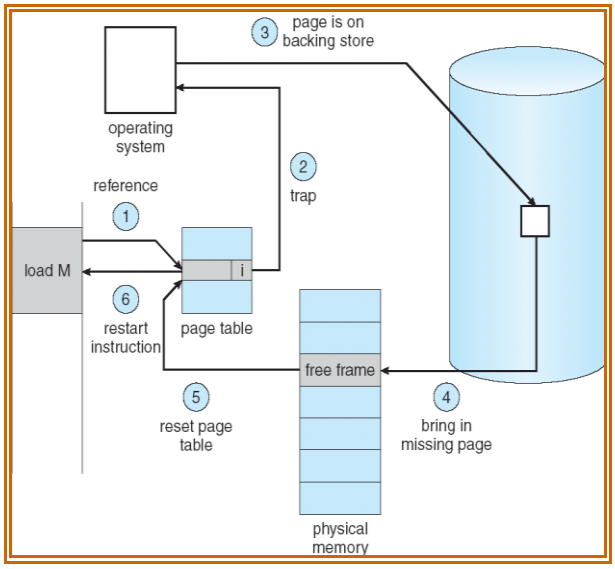
Falha de página



- Se houver uma referência a uma página, a primeira referência vai provocar a chamada do sistema operacional => falha de página
- O sistema operacional olha para uma outra tabela para decidir se:
  - a referência é inválida => aborta o processo
  - Apenas não está na memória
- Obtem-se um quadro livre
- Coloca (*swap*) a página no quadro
- Atualiza as tabelas, bit de válido-inválido=1
- Volta a executar a instrução que foi interrompida.

E se não houver quadros livres?

- Substituição de páginas - achar uma página na memória, mas que não esteja em uso, retirá-la (*swap it out*)



  - Algoritmo de troca
  - Desempenho - procura-se um algoritmo que resulte no menor número de falhas de páginas
- A mesma página pode ser trazida para a memória diversas vezes.

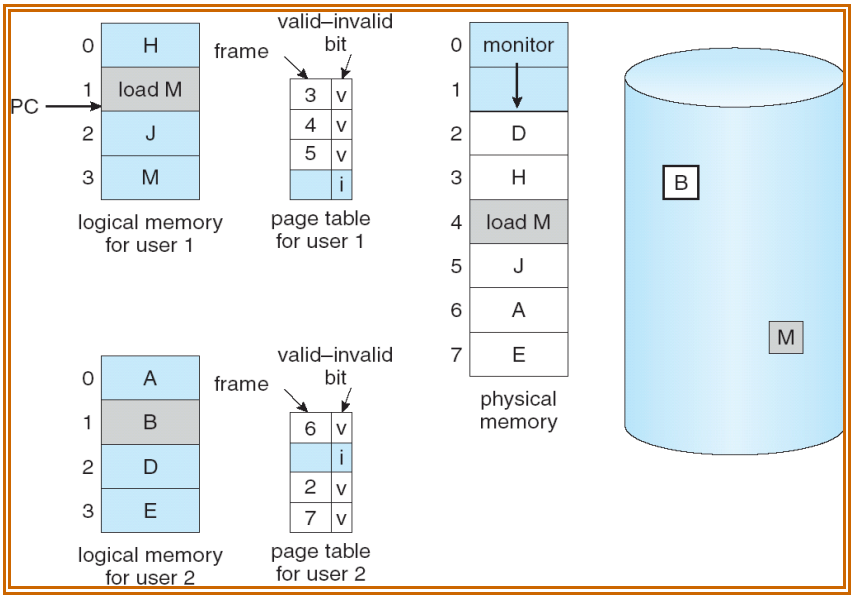
Desempenho da paginação por de demanda pura

A seção 10.2.2 do livro ilustra um cálculo de como a paginação afeta o tempo de execução de um processo. O importante a reter é a conclusão de que para permitir uma redução de velocidade de 10% é necessário que a taxa de falha de páginas seja menor do 1 para cada 2.500.000 acessos à memória. Isto se deve ao alto custo que uma falha provoca, pois envolve uma busca de página no disco que é muito lento em relação à memória principal.

Recomendo que os alunos com maior habilidade com números estudem esta seção e chequem a validade dos cálculos.

Substituição de páginas

- Adapta-se ao *overbooking* de páginas de memória num ambiente de multiprogramação, a rotina de serviço que trata a falha de páginas, trata também a substituição de páginas.
- Um bit de *modificação (dirty)* é acrescentado para reduzir a quantidade de transferências, apenas as páginas *sujas* são escritas de volta no disco.
- A substituição de páginas completa a separação lógica entre memória lógica e física - um grande espaço de memória virtual pode ser oferecido por uma pequena memória física.



Algoritmos de substituição

- Procura-se o mais baixa taxa de falhas de página.
- Avalie-se o algoritmo pela execução de uma determinada seqüência de referências à memória, geralmente simplificada para uma seqüência de referências às páginas diferentes de memória e calcula-se o número de falhas de páginas para esta seqüência.
- Nos exemplos a seguir, a seqüência de referências será:

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

Algoritmo Primeiro a chegar, Primeiro a sair (FIFO)

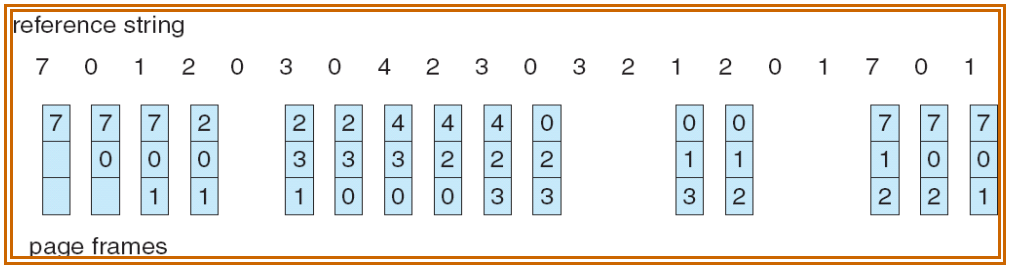
3 quadros (apenas 3 páginas na memória de cada vez para cada processo)

1	1	4	5	
2	2	1	3	9 faltas de página
3	3	2	4	

4 quadros

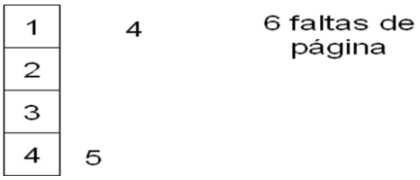
1	1	5	4	
2	2	1	5	10 faltas de página
3	3	2		
4	4	3		

- Substituição por FIFO - provoca anomalia de Belady
  - mais quadros não implicam em menos falhas de páginas.

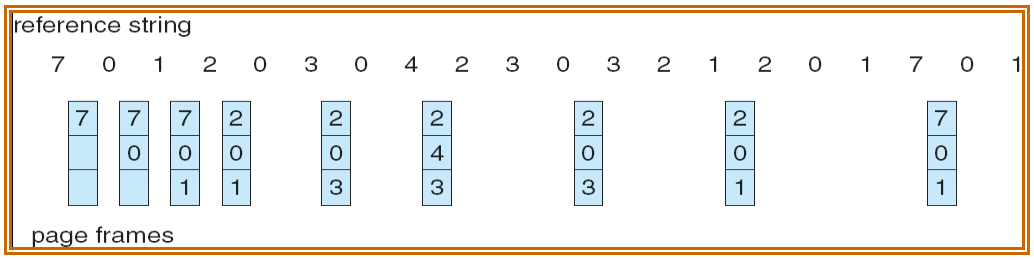


Algoritmo ótimo

- Substituir a página que vai demorar mais para voltar a ser usada.
- Exemplo com 4 quadros

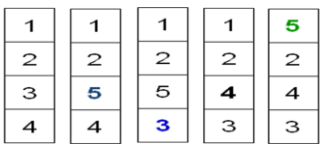


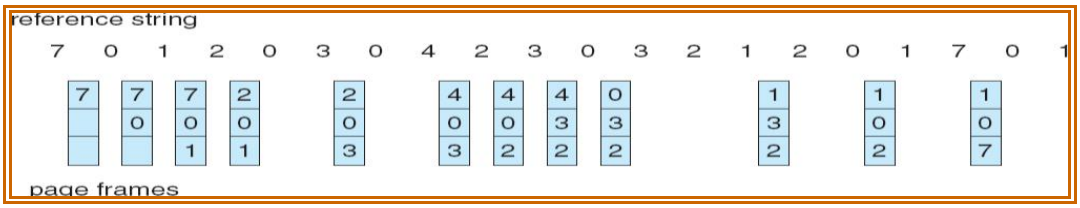
- Como descobrir esta página?
- Usado para determinar quão bom é um algoritmo em relação a este ótimo.



Algoritmo Usado Menos Recentemente (LRU)

- 4 quadros





- Implementação com um contador
    - Cada linha da tabela de páginas tem um contador, cada vez que uma referência é feita a esta página, o relógio é copiado para este contador.
    - Quando uma página precisa ser trocada, olha-se nos contadores para determinar qual trocar.
- Implementação usando uma pilha - mantem-se uma pilha com o número das páginas numa lista duplamente encadeada:
    - Referência a uma página
        - Mova-a para o topo da pilha
        - 6 ponteiros devem ser trocados
    -

É necessário suporte de *hardware* para a implementação do LRU, o uso de software implicaria que cada acesso à memória provocaria a execução de diversas instruções e isto não seria aceitável. Poucas arquiteturas tem um suporte completo para implementar o LRU.

Algoritmos aproximativos de LRU

- Bit de referência
    - Cada página tem associada um bit, inicialmente =0
    - Quando uma página é referenciada, o bit =1
    - Substitua aquela com bit =0 (se existir). Em qualquer ordem.
- Segunda chance
    - Precisa do bit de referência
    - Substituição no sentido horário
    - Se a página a ser substituída (no sentido horário) tiver o bit de referência =1, então:
        - coloque o bit de referência em 0
        - deixe a página na memória
        - tente substituir a próxima página (no sentido horário), usando as mesmas regras.

Algoritmos de contagem

- Mantenha um contador para o número de referências de cada página
- Algoritmo LFU: substitui a página com menor frequência
- Algoritmo MFU: baseado no argumento de que uma página recentemente trazida tem o menor número de referências e ainda precisa ser usada.