Ministério da Educação
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
UNED Nova Friburgo
Bacharelado em Sistemas da Informação

Gerência de Memória (2)

Sistemas Operacionais

Prof. Bruno Policarpo Toledo Freitas
bruno.freitas@cefet-rj.br

Objetivos

- **Compreender as faltas de páginas e suas consequências no desempenho de sistemas de memória**
- **Comparar as vantagens e desvantagens de diferentes algoritmos para faltas de página**
- **Apresentar como a paginação é implementada e usada nos sistemas operacionais**
- **Compreender como a implementação da paginação pode afetar o desempenho do sistema.**
- **Introduzir a segmentação**

Algoritmos de substituição de páginas

- **Falta de páginas força escolha:**
  - Página que precisa ser removida
  - Liberar espaço para página nova
- **Página modificada precisa ser salva**
  - Páginas não modificadas apenas são sobreescritas
- **Melhor não escolher páginas usadas recentemente**
  - Provavelmente deverão ser trazidas de novo
  - Algoritmo ótimo: escolhe a página que será utilizada no ponto mais distante no futuro

Algoritmos de substituição de páginas
Não Usada Recentemente (NRU)

- **Cada página da tabela de páginas possui:**
  - 1 bit de referência (R)
  - 1 bit de modificada (M)
- **Páginas são classificadas em:**

  1)Não referenciada, não modificada

  2)Não referenciada, modificada

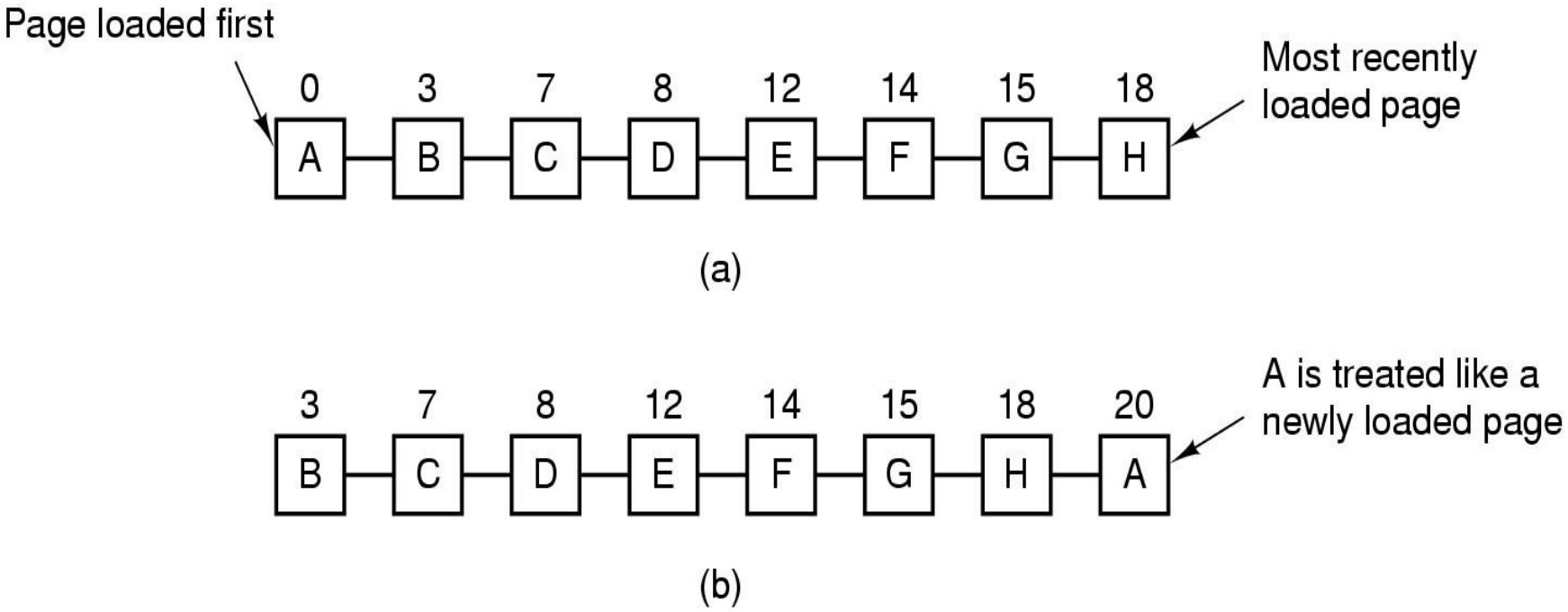
  3)Referenciada, não modificada

  4)Referenciada, modificada
- **NRU remove página randômica do menor ao maior**

Algoritmos de substituição de páginas
FIFO

- **Mantém uma lista de todas as páginas na memória**
  - Na ordem com que elas chegam
- **Página no início da fila é a que será substituída**
- **Desvantagem:**
  - Página na memória por mais tempo pode ser a mais referenciada

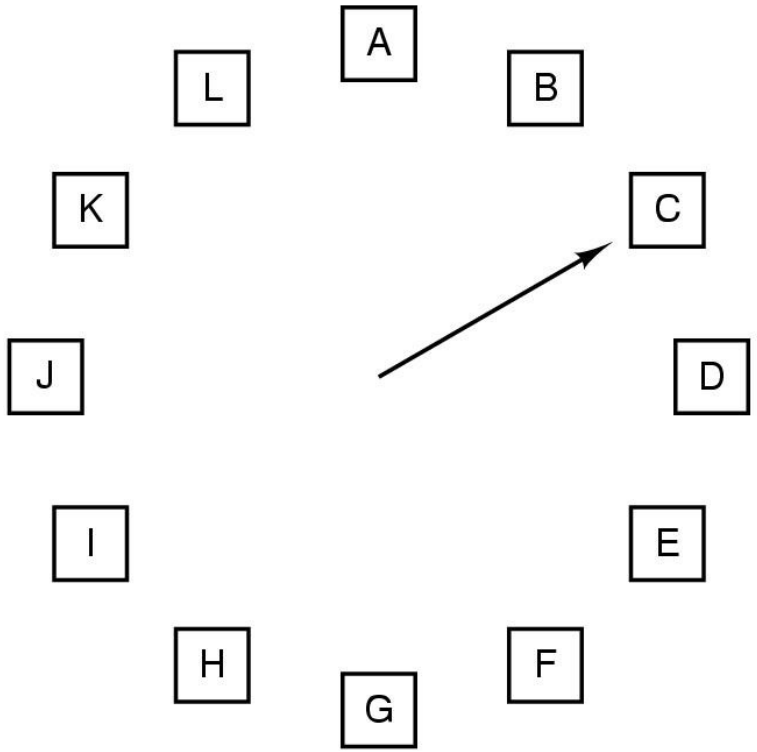
Algoritmos de substituição de páginas
Segunda Chance



- **Páginas organizadas como a FIFO**
- **Bit “R” indica se a página foi usada (R=1)**
- **Se sim, R=0 e volta ao fim da fila**

Algoritmos de substituição de páginas
Algoritmo do Relógio

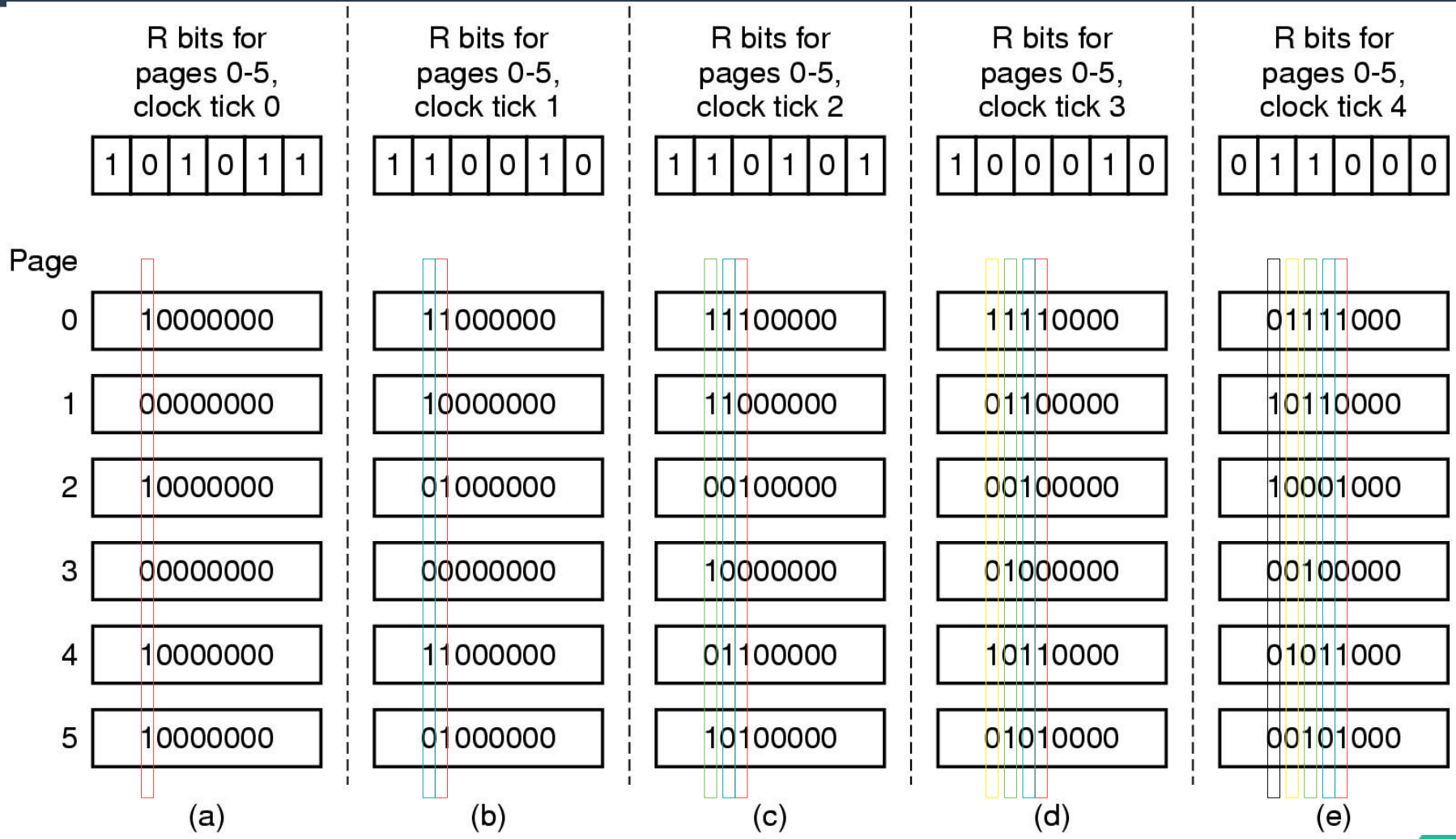
- **FIFO com segunda chance mais eficiente**
- **Quando uma falta ocorre, ação depende de onde está o ponteiro:**
  - R=0: Tira a página
  - R=1: R← 0, avança ponteiro e repete



Algoritmos de substituição de páginas
Menos Usada Recentemente (LRU)

- **Assume que páginas usadas recentemente serão reutilizadas novamente**
  - Descarta página que está sem ser utilizada por mais tempo
  - Necessário: contar referências à memória
- **Implementação precisaria de hardware especial**
  - Hardware contém um “contador de referências” C
  - Cada entrada da tabela de páginas contém seu próprio contador de referências
  - Cada referência à memória salva C na respectiva entrada da tabela
  - Em falta de página: escolhe página com o menor contador
  - Impraticável: somadores são lentos

Algoritmos de substituição de páginas
Envelhecimento



Exercício

Considere um computador com 4 molduras de página usando o algoritmo do envelhecimento

Considere que os bits R de cada página em 5 interrupções de relógio são: 1001 – 1100 – 1111 – 0001 – 0100 e o registrador de tempo tenha 4 bits.

Se ocorrer uma falta de página após a última interrupção, qual página será retirada da memória?

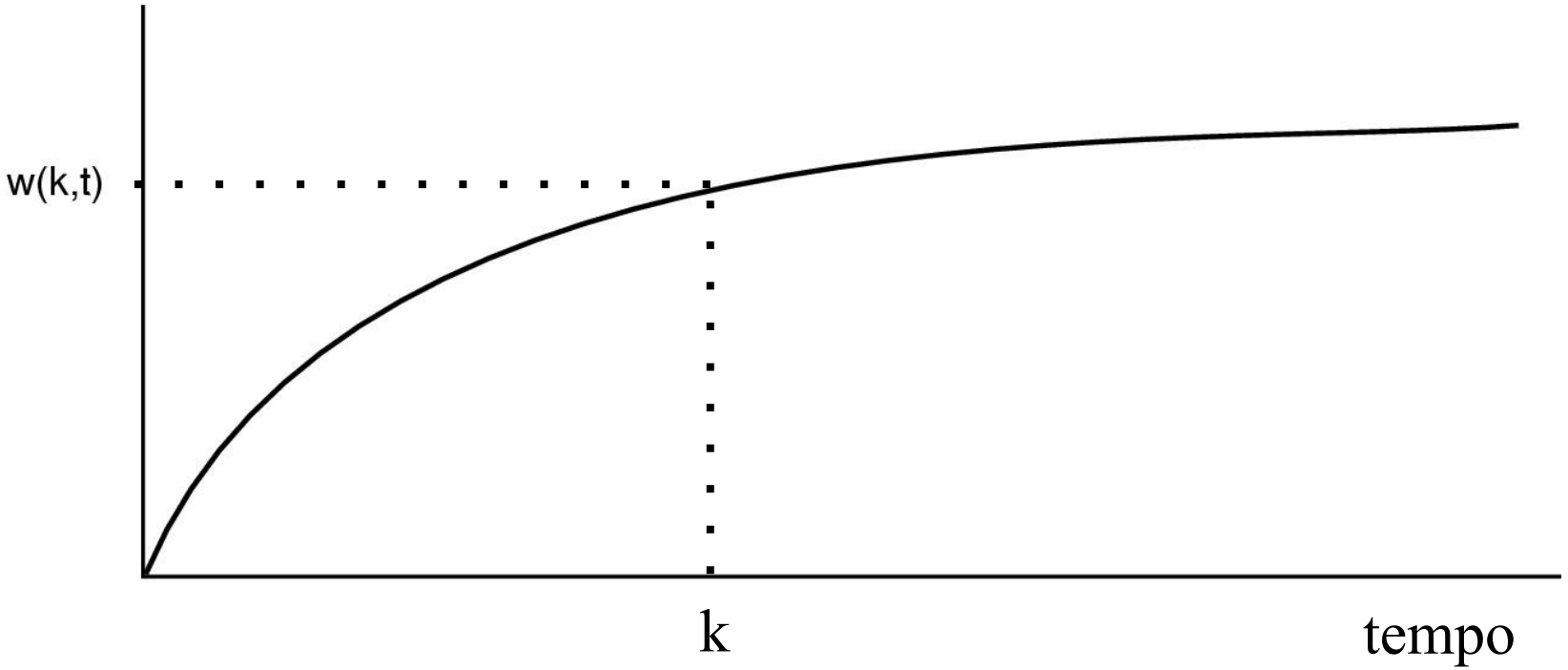
Algoritmos de substituição de páginas
Conjunto de Trabalho

- **Na forma mais pura de paginação, processos são inicializados sem qualquer página na memória**
  - Paginação sob demanda: páginas vão sendo carregadas conforme necessidade.
  - Funciona porque processos em geral apresentam a localidade de referência.
  - O conjunto de páginas que um processo está utilizando naquele momento é chamado de conjunto de trabalho.

Algoritmos de substituição de páginas
Conjunto de Trabalho

- **Se a memória for curta a ponto do conjunto de trabalho de um processo não couber, deverá ser aberto espaço na memória:**
  - Páginas são retiradas e postas de volta
  - *Thrashing*

Algoritmos de substituição de páginas
Modelagem do Conjunto de Trabalho

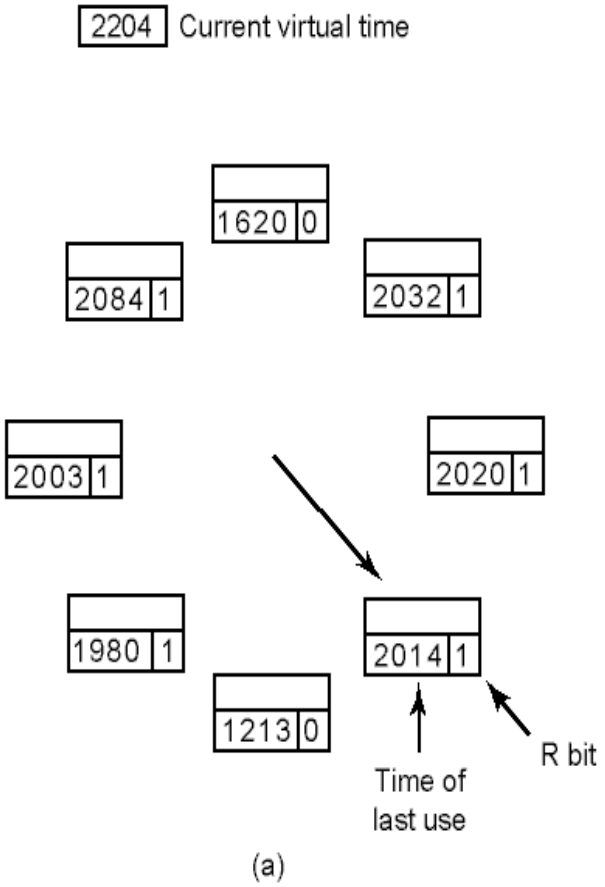


- **Para um dado instante de tempo *t*, o conjunto de trabalho w(k,t) é o conjunto de páginas utilizadas pelas k-últimas referências à memória**
- **Melhor algoritmo é o WSCLOCK**

Algoritmos de substituição de páginas
WSClock

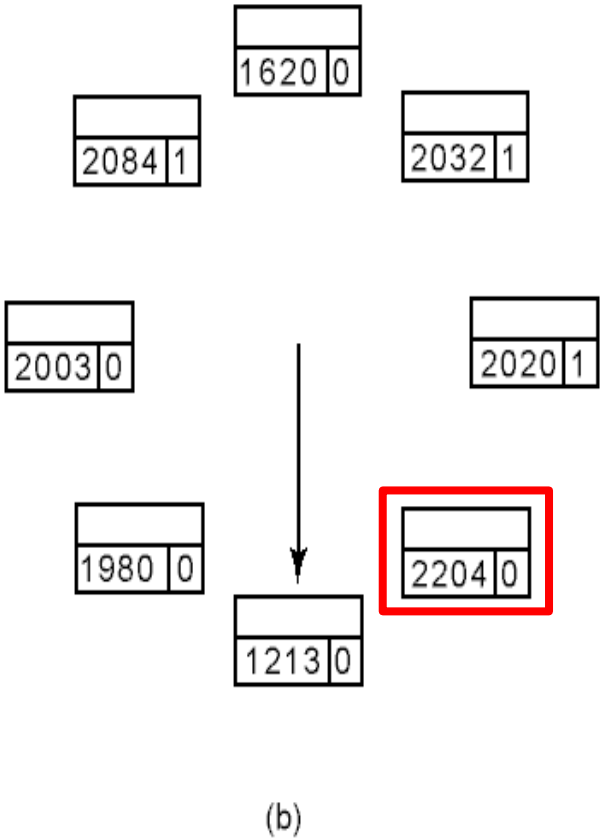
- **Idéia semelhante ao Clock mas levando em conta o conjunto de trabalho dos últimos τ *segundos virtuais***
- **Cada página possui:**
  - Bit *M*
  - Bit *R*
  - ***Time of last use***

WSClock
Exemplo 1



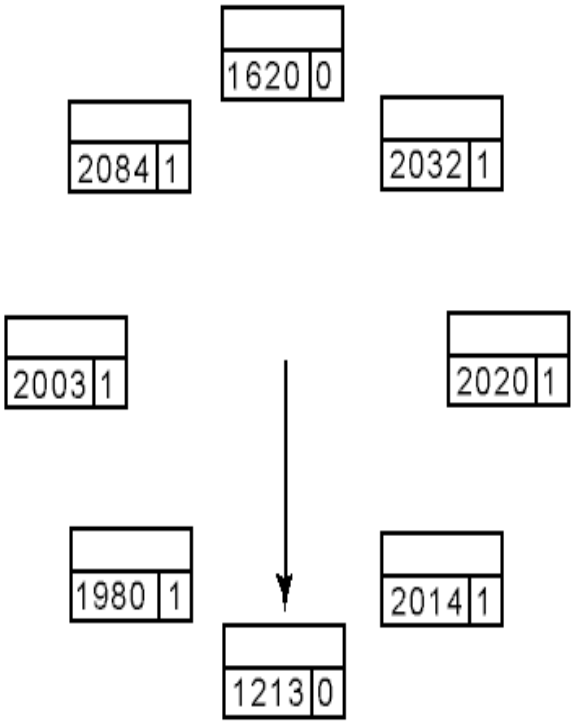
- **Considerando:**
  - τ = 200
  - Falta de página no tempo virtual 2204
  - Ponteiro para a página acessada no tempo 2014
- **Página referenciada e dentro do τ**
  - Avança ponteiro
  - Atualiza o tempo virtual da página
  - Zera o R (R = 0)

WSClock
Exemplo 1



- **Considerando:**
  - τ = 200
  - Falta de página no tempo virtual 2204
  - Ponteiro para a página acessada no tempo 2014
- **Página referenciada e dentro do τ**
  - Avança ponteiro
  - Atualiza o tempo virtual da página
  - Zera o R (R = 0)

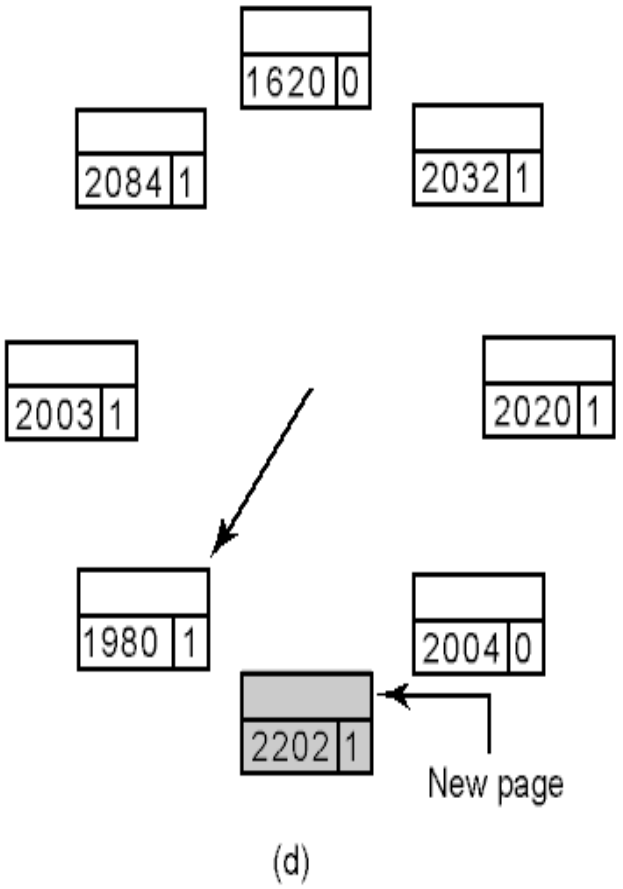
WSClock
Exemplo 2



(c)

- **Considerando:**
  - τ =200
  - Falta de página no tempo 2202
  - Ponteiro na página acessada no tempo 1213
- **Não referenciado e fora do τ**
  - Substitui para nova página
  - Tempo virtual igual ao tempo da falta

WSClock
Exemplo 2



- **Considerando:**
  - τ =200
  - Falta de página no tempo 2202
  - Ponteiro na página acessada no tempo 1213
- **Não referenciado e fora do τ**
  - Substitui para nova página
  - Tempo virtual igual ao tempo da falta

Exercício

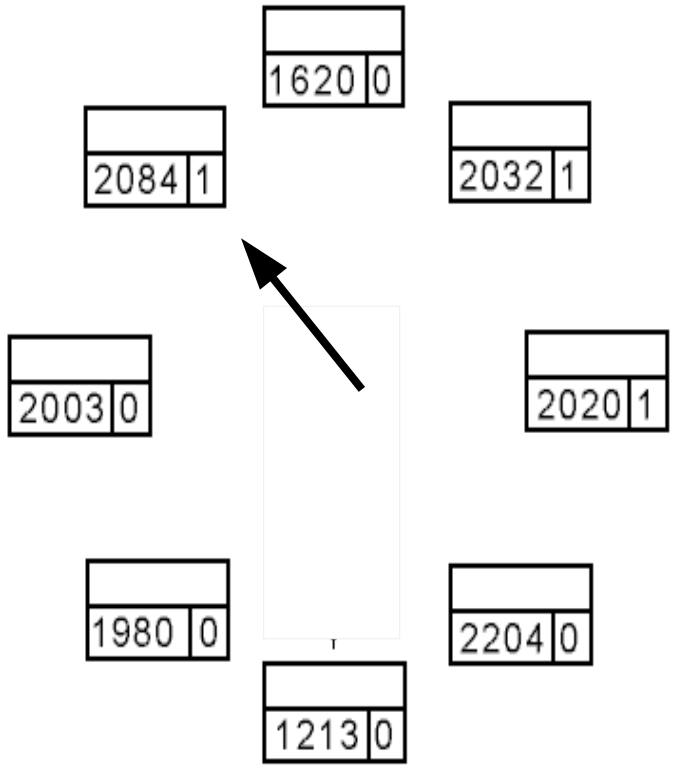
Suponha o algoritmo WSClock na situação ao lado.

Considerando o tempo virtual como 2300, qual página será retirada quando ocorrer uma falta de página considerando os seguintes τ:

a)300

b)200

c)1000



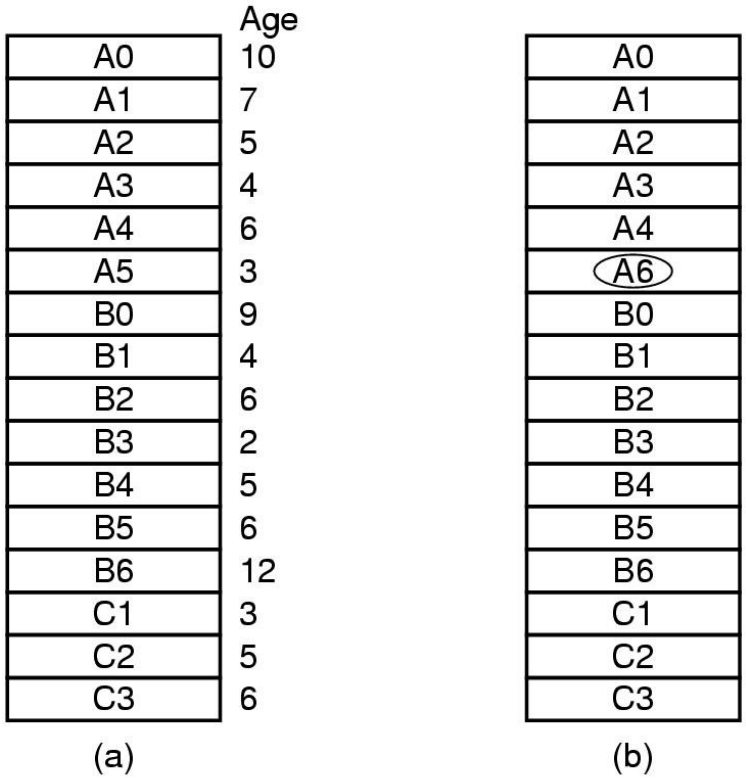
(b)

Algoritmos de Substituição de Páginas
Sumário

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

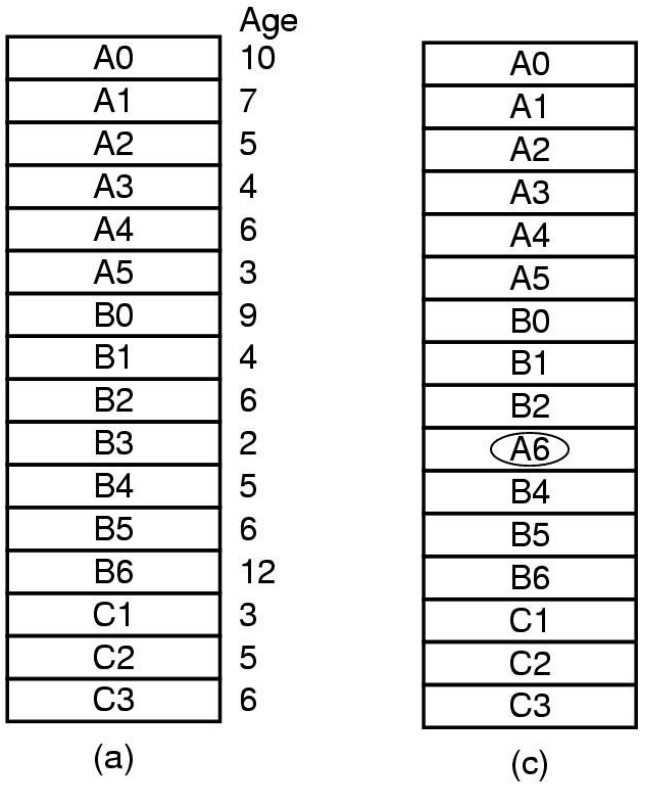
Design de sistemas de paginação
Política de substituição local

- **Seleciona página dentre as aquelas alocadas apenas para o processo**

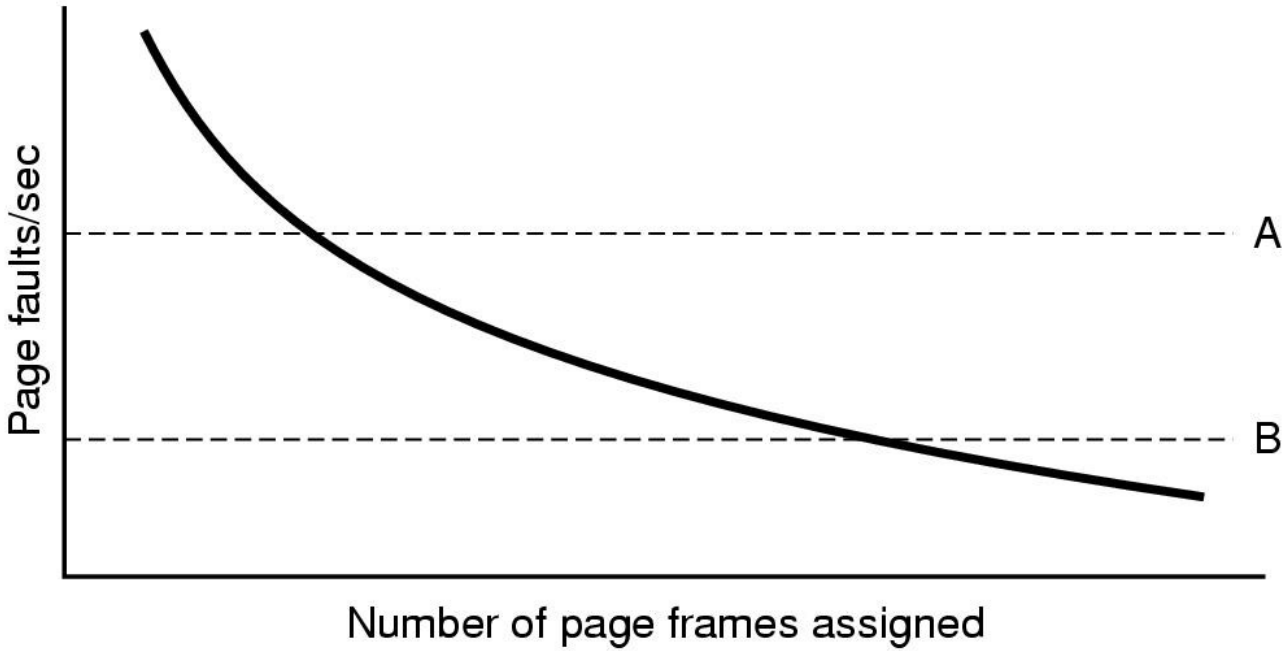


Design de sistemas de paginação
Política de substituição global

- **Seleciona página dentre todos os processos**
- **Geralmente possui melhor desempenho**
- **Permite adaptação dinâmica do tamanho das páginas do processo**



Algoritmo *Page Fault Frequency*



- **Usado em políticas de substituição global de páginas**
- **Estabele quantidade de páginas alocadas para um processo**
  - Conta-se a taxa de faltas
  - Aumenta/diminui páginas de acordo

Controle de carga

- **Mesmo com bons designs, sistemas podem entrar em *thrashing***
  - Conjunto de trabalho dos processos excede capacidade do sistema
- **Quando no algoritmo PFF ...**
  - Alguns processos precisam de mais memória
  - ... mas nenhum processo precisa de menos
- **Solução: reduzir o número de processos competindo pela memória**
  - Trazer um ou mais processos para o disco, dividindo as páginas que eles possuem
  - Reconsiderar o grau de multiprogramação

Tamanho de página

- **Em geral, páginas pequenas são melhores:**
  - Menor fragmentação interna
  - Adapta-se melhor a diferentes estruturas e tamanhos de seções de códigos, dados e pilha
  - Mais programas podem estar na memória
- **Porém, páginas menores implicam em:**
  - Desperdício da TLB
  - Tabela de páginas maiores

Tamanho de página
Overhead da paginação

- **Ocorre devido à tabela de páginas e fragmentação interna**
- **Seja:**
  - s = tamanho médio dos processos (bytes)
  - p = tamanho da página (bytes)
  - e = entrada de uma tabela de páginas

$$overhead = \frac{s \cdot e}{p} + \frac{p}{2}$$

Tamanho da tabela

Fragmentação interna (estimativa)

Ótimo quando:

$$p = \sqrt{2se}$$

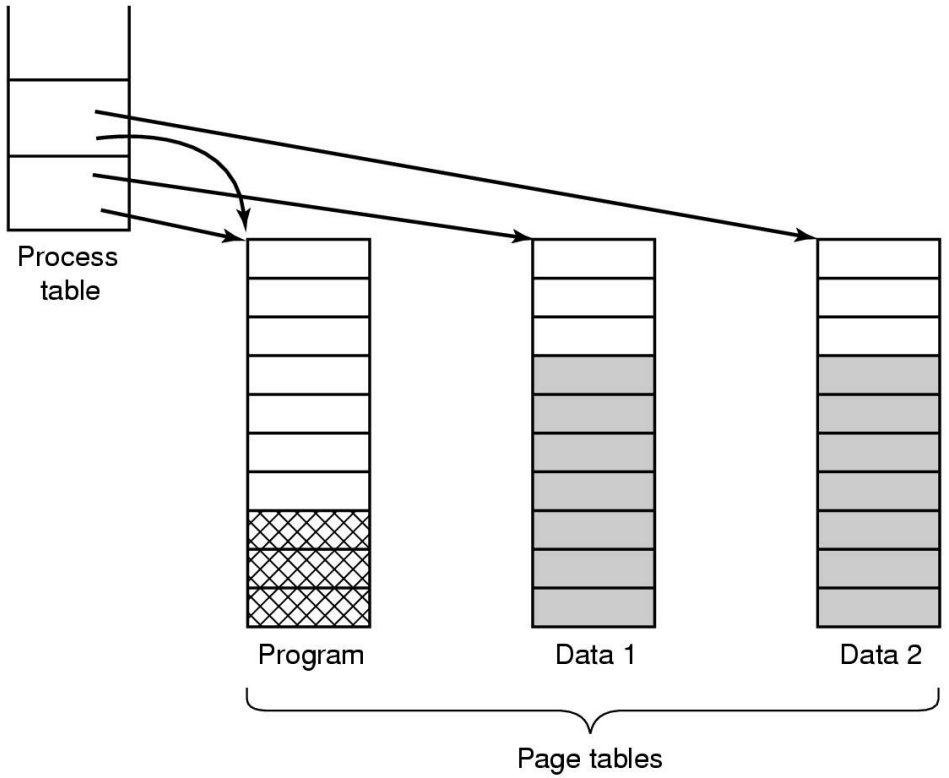
Exercício

- **Suponha um processo com 1,3 MiB de tamanho em um sistema de paginação em que a página possui 4 KiB de tamanho.**

  1) Quantas páginas esse processo ocupa?
  2) Qual é o tamanho de memória perdido no sistema devido a fragmentação interna esse processo?

Páginas compartilhadas

- **Múltiplos usuários podem estar executando um mesmo programa**
- **Chamadas fork() de processos não criam páginas de código nova**
- **Páginas de dados e pilha iguais porém com *copy-on-write* (COW)**



Artigo *DLL Hell*

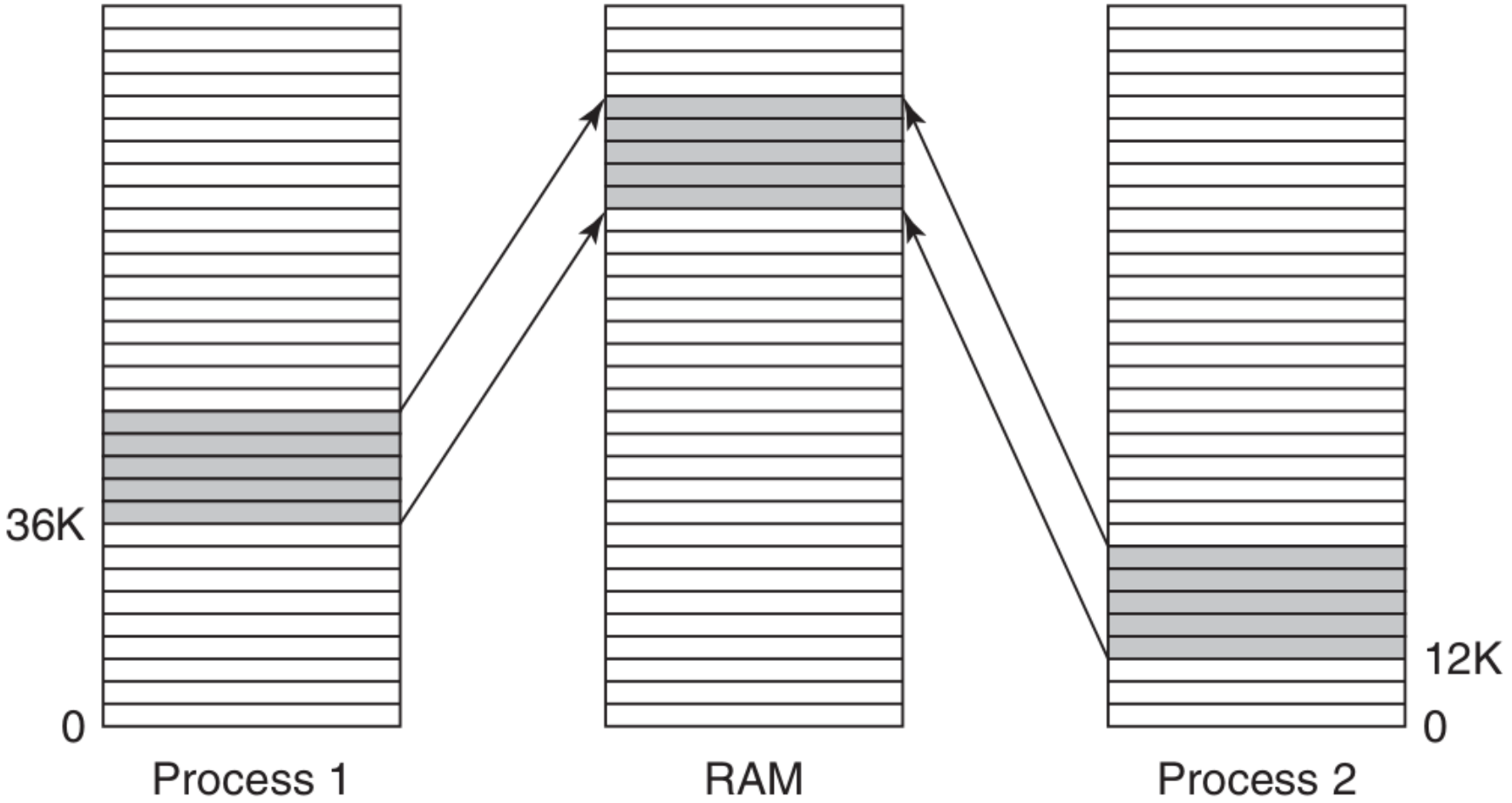
Páginas compartilhadas
Bibliotecas compartilhadas

- **Páginas compartilhadas são utilizadas para a implementação de bibliotecas compartilhadas.**
  - .dll no Windows e .so no UNIX
  - Evitam religações desnecessárias
  - Facilitam atualização de programas dependentes
  - Diminuem espaço de armazenamento geral
- **Necessitam de código *independente de posição***
  - gcc -shared lib.o -fPIC

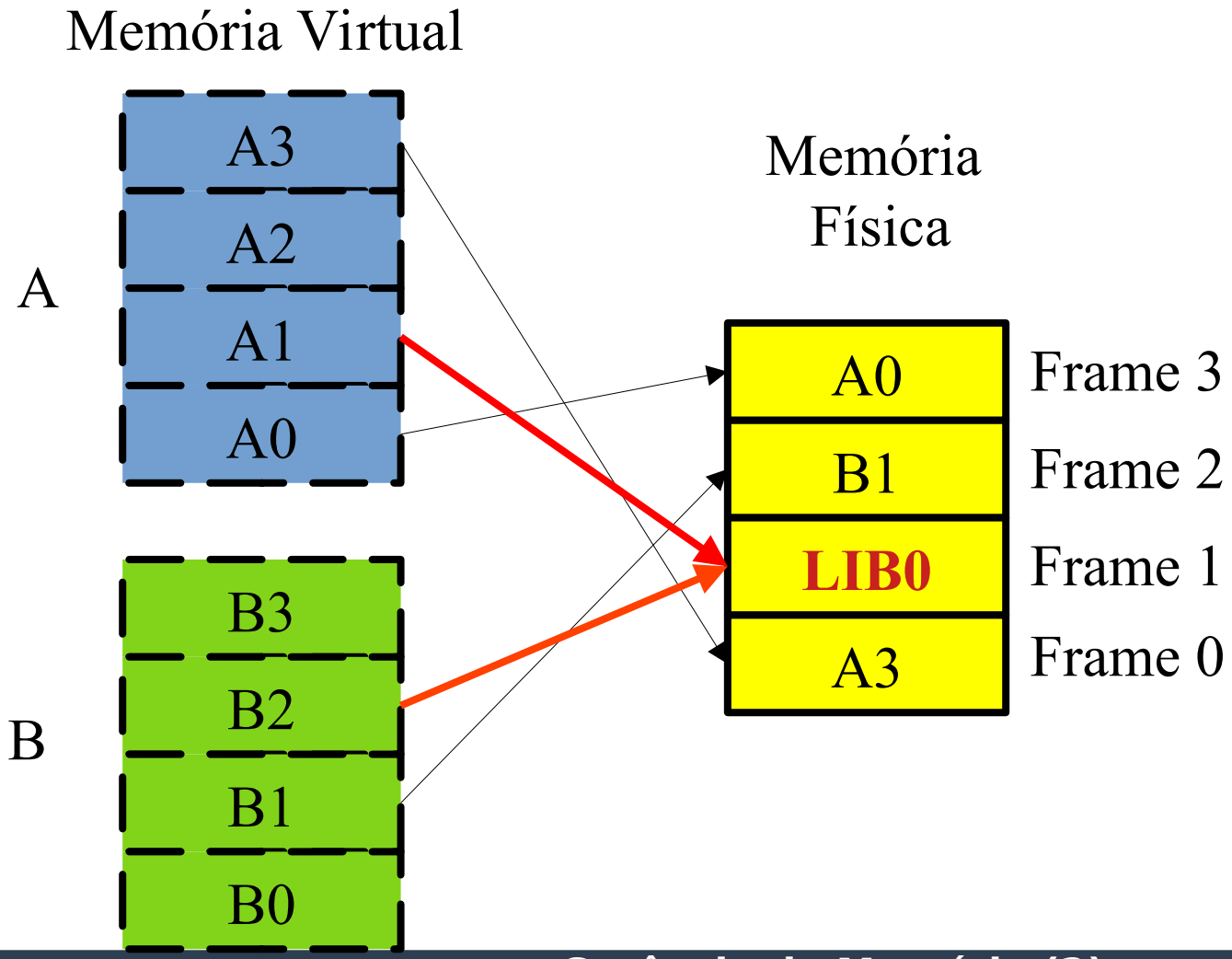
Páginas compartilhadas

- **Exemplo de biblioteca compartilhada ...**

Bibliotecas compartilhadas
Código Independente de Posição



Bibliotecas compartilhadas
Código Independente de Posição



Arquivos mapeados em memória

- **Semelhante à páginas compartilhadas**
- **Arquivos podem ter porções mapeadas na memória**
  - Vetor de caracteres
  - Modelo alternativo aos *read()*/*write()*
  - “Desmapear” o arquivo o salva na memória secundária
- **Modelo “cru” de Comunicação Entre Processos**

Política de limpeza

- **Paginação funciona melhor quando há frames livres**
- **Processo daemon em background**
  - Periodicamente inspeciona o estado da memória
  - Quando poucos frames estão livres, seleciona páginas para retirar usando algum algoritmo de substituição
- **Pode usar algoritmo diferente da falta de páginas**
  - WSClock

Questões de implementação
Atuação do Sistema Operacional

1) Criação de um novo processo:

- Estimar tamanho inicial do processo
- Alocar tabela de páginas e páginas iniciais
- Alocar espaço na área de *swap*
- Informações da tabela de páginas e da área de swap devem ser salvas no descritor de processo

Questões de implementação
Atuação do Sistema Operacional

2) Execução do processo:

- MMU e TLB recebem um *reset*
- Tabela de páginas do processo atual é carregada
- Opcional: carregar páginas do processo

Questões de implementação
Atuação do Sistema Operacional

3) Falta de página:

- Fazer backup do Contador de Programa que gerou a falta
- SO determina qual endereço virtual gerou a falta
- Computar a página necessária, liberar/encontrar uma moldura e colocá-la lá

Questões de implementação
Atuação do Sistema Operacional

4) Término de processo:

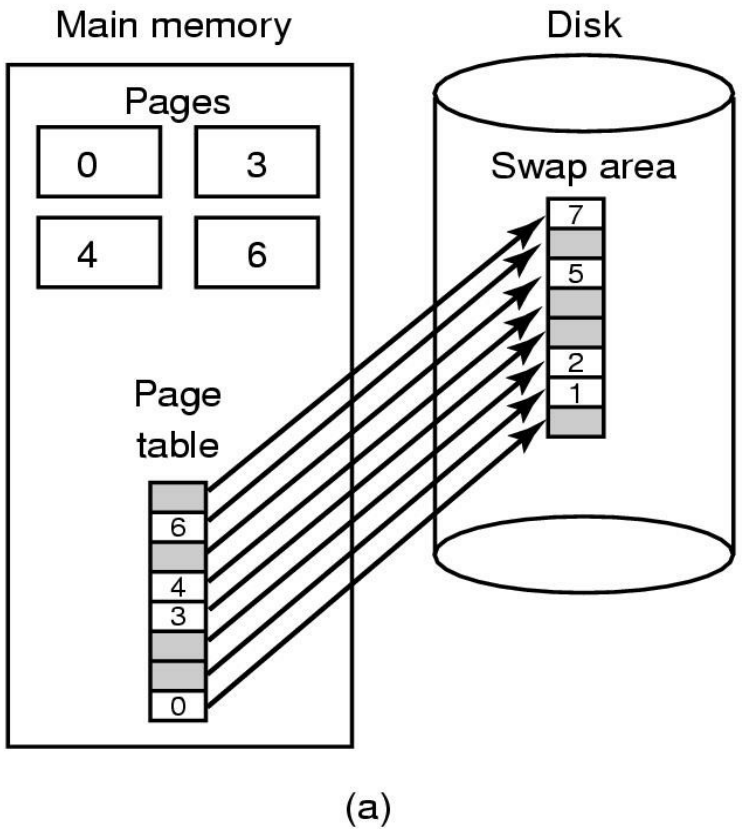
- Libera páginas alocadas na memória
- Libera tabela de páginas
- Libera espaço no disco

Retenção de páginas

- **Memória virtual e I/O ocasionalmente interagem**
- **Processo pode realizar um read**
  - Enquanto espera I/O, outro processo inicia
  - Falta de página
  - Buffer de I/O do primeiro processo pode ser escolhido
- **Necessário travar algumas páginas**

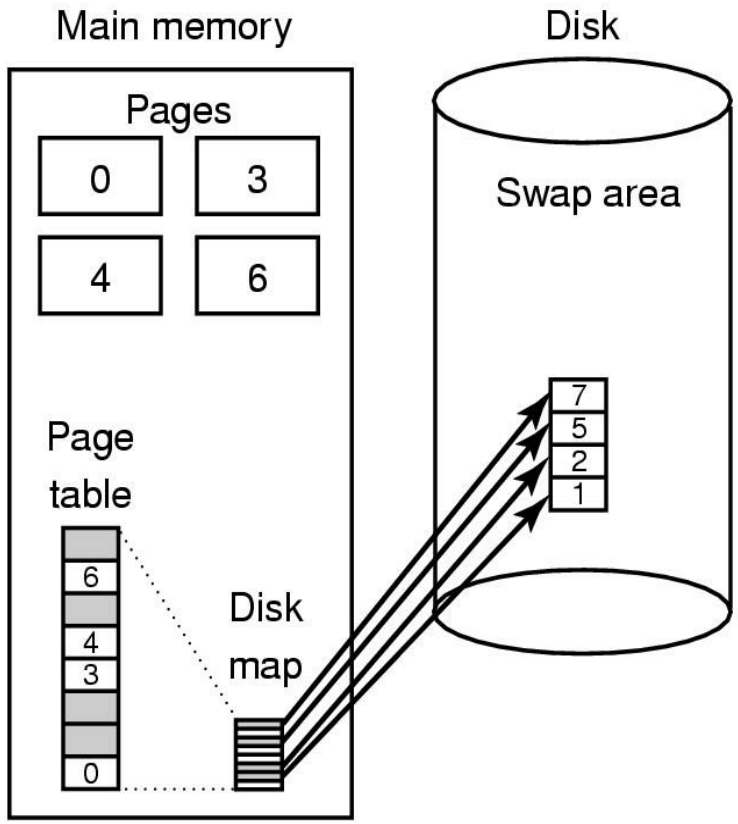
Memória secundária

- **Área estática:**
  - Relação 1x1 entre lugar na memória principal e disco
  - Pressupõe cópia das páginas no disco



Memória secundária

- **Área dinâmica:**
  - Página em disco é eliminada para nova
  - Mapa de disco contém os endereços das páginas no disco
  - Mapa de disco é atualizado



Memória secundária
Localização de páginas no disco

- **Partição (UNIX)**
  - Vantagem: elimina *overhead* de arquivo
  - Desvantagem: tamanho fixo
- **Arquivo (Windows)**
  - Vantagem: tamanho variável/ajustável
  - Desvantagem: utiliza infraestrutura de I/O
  - Distribuições recentes UNIX implementam esse esquema por padrão

Gerência de memória no Linux

Aplicativo/arquivo	Descrição
free	Ocupação de memória
pmap PID	Mapeamentos de memória do processo
/proc/pid/pagemap	Mapeamentos de página do processo
/proc/kpageflags	Flags das página na memória

Estudo de caso: Linux

- **https://www.kernel.org/doc/html/latest/admin-guide/mm/pagemap.html**
- **Dois arquivos principais:**
  - */proc/pid/pagemap*
    - Informações sobre as páginas, indexadas pela página
  - */proc*/pid/kpageflags
    - Propriedades das páginas, indexadas pela páginas
- **Arquivos podem ser lidos somente se a flag de compilação do kernel CAP_SYS_ADMIN estiver habilitada**

Estudo de caso: Linux

- `/proc/pid/pagemap`. This file lets a userspace process find out which physical frame each virtual page is mapped to. It contains one 64-bit value for each virtual page, containing the following data (from `fs/proc/task_mmu.c`, above pagemap_read):
    - Bits 0-54 page frame number (PFN) if present
    - Bits 0-4 swap type if swapped
    - Bits 5-54 swap offset if swapped
    - Bit 55 pte is soft-dirty (see Soft-Dirty PTEs)
    - Bit 56 page exclusively mapped (since 4.2)
    - Bit 57 pte is uffd-wp write-protected (since 5.13) (see Userfaultfd)
    - Bits 58-60 zero
    - Bit 61 page is file-page or shared-anon (since 3.5)
    - Bit 62 page swapped
    - Bit 63 page present

Estudo de caso: Linux

- `/proc/kpageflags`. This file contains a 64-bit set of flags for each page, indexed by PFN.

  The flags are (from `fs/proc/page.c`, above kpageflags_read):

  0. LOCKED
  1. ERROR
  2. REFERENCED
  3. UPTODATE
  4. DIRTY
  5. LRU
  6. ACTIVE
  7. SLAB
  8. WRITEBACK
  9. RECLAIM
  10. BUDDY
  11. MMAP
  12. ANON
  13. SWAPCACHE
  14. SWAPBACKED
  15. COMPOUND_HEAD
  16. COMPOUND_TAIL
  17. HUGE
  18. UNEVICTABLE
  19. HWPOISON
  20. NOPAGE
  21. KSM
  22. THP
  23. OFFLINE
  24. ZERO_PAGE
  25. IDLE
  26. PGTABLE

Estudo de caso: Linux

- **https://www.kernel.org/doc/html/latest/admin-guide/mm/multigen_lru.html**

  Maiores informações sobre o funcionamento do conjunto de trabalho

Estudo de caso: Linux

LF live MENTORSHIP SERIES

Intel Processor TLB Sizes

Processor	ITLB 4K	ITLB 2M	DTLB 4K	DTLB 2M	DTLB 1G	STLB 4K	STLB 2M	STLB 1G
Nehalem	128	7	64	32	4	512	-	-
Sandy Br.	128	8	64	32	4	512		
Haswell	128	8	64	32	4	1024		
Sky Lake	128	8	64	32	4	1536		16
Ice Lake						2048	1024	1024

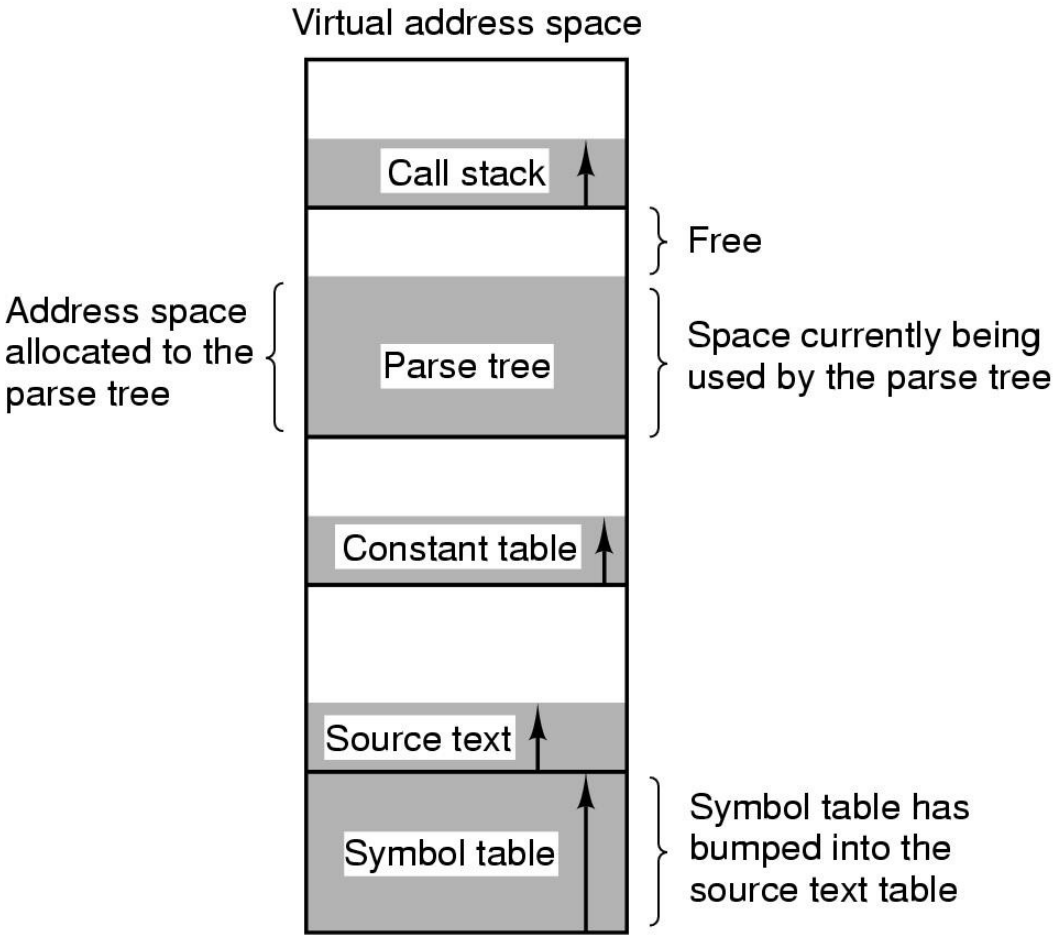
Estudo de caso: Linux

- **Outras características:**
  - Ferramenta de análise e otimização de memória (DAMON)
  - Suporte a páginas grandes (HugePages)
  - Permite verificar uso de páginas de uma carga de trabalho (Idle Page Tracking)
  - Compactação de memória
  - Possível fazer gerência de memória virtual em espaço de usuário (*userfaultfd*)
  - Páginas marcadas para swap primeiramente são compactadas em uma região de memória específica (*zswap*)
  - Parâmetro *swappiness* para definir quando o sistema se preocupara em realizar *swap-out* de páginas

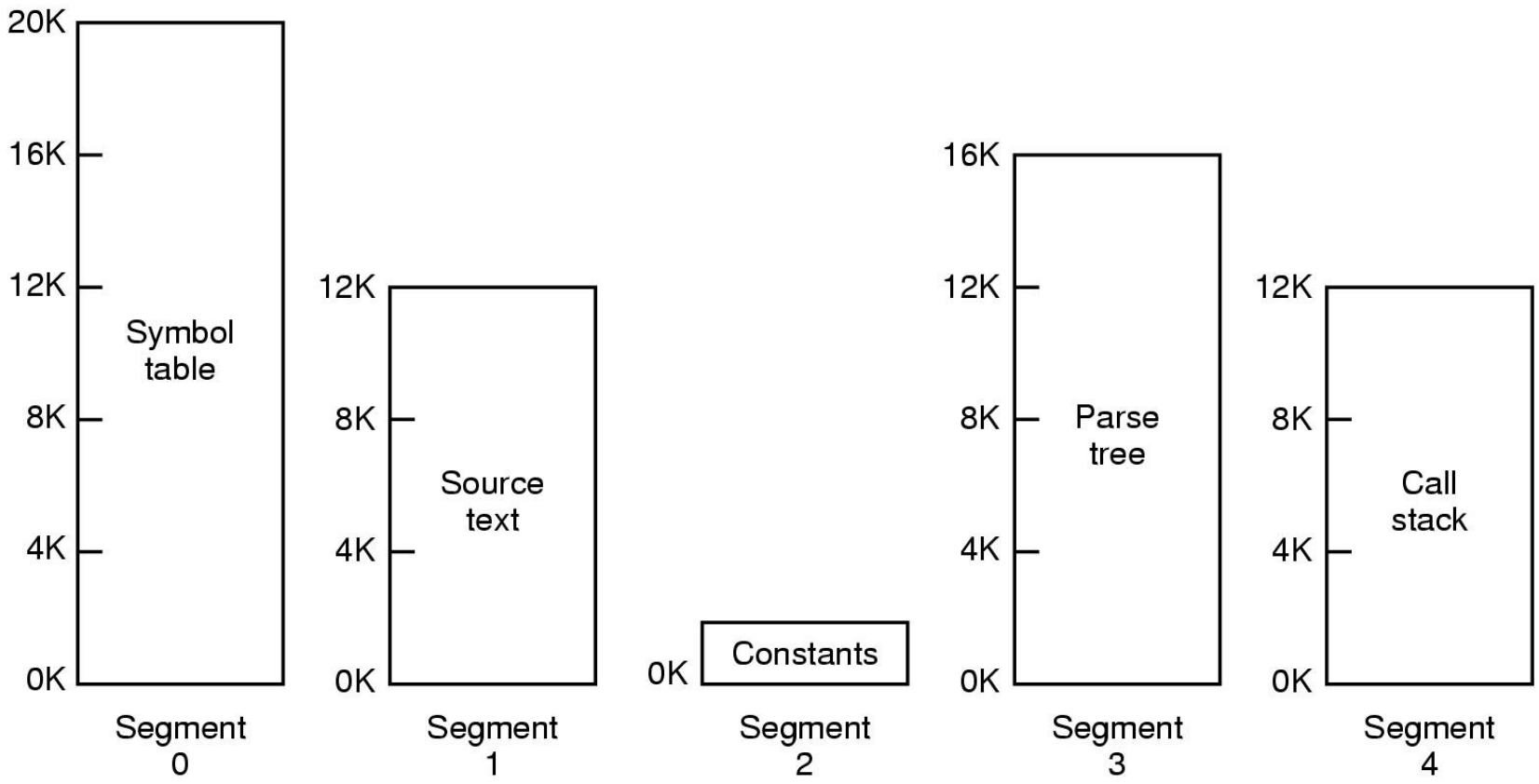
Segmentação

- **Memória é dividida em vários espaços de endereçamento isolados uns dos outros**
- **Criado pela Intel para contornar limitações tecnológicas da época**
  - Adição de lógica adicional e registradores no chip
- **Considerada obsoleta desde a arquitetura x86-64, mantida apenas para fins de compatibilidade**

Segmentação
Espaço de Endereçamento único



Segmentação
Espaços de endereçamentos segmentados



Permite que cada tabela cresça ou diminua independentemente

Comparação

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Gerência de Memória no IA-32

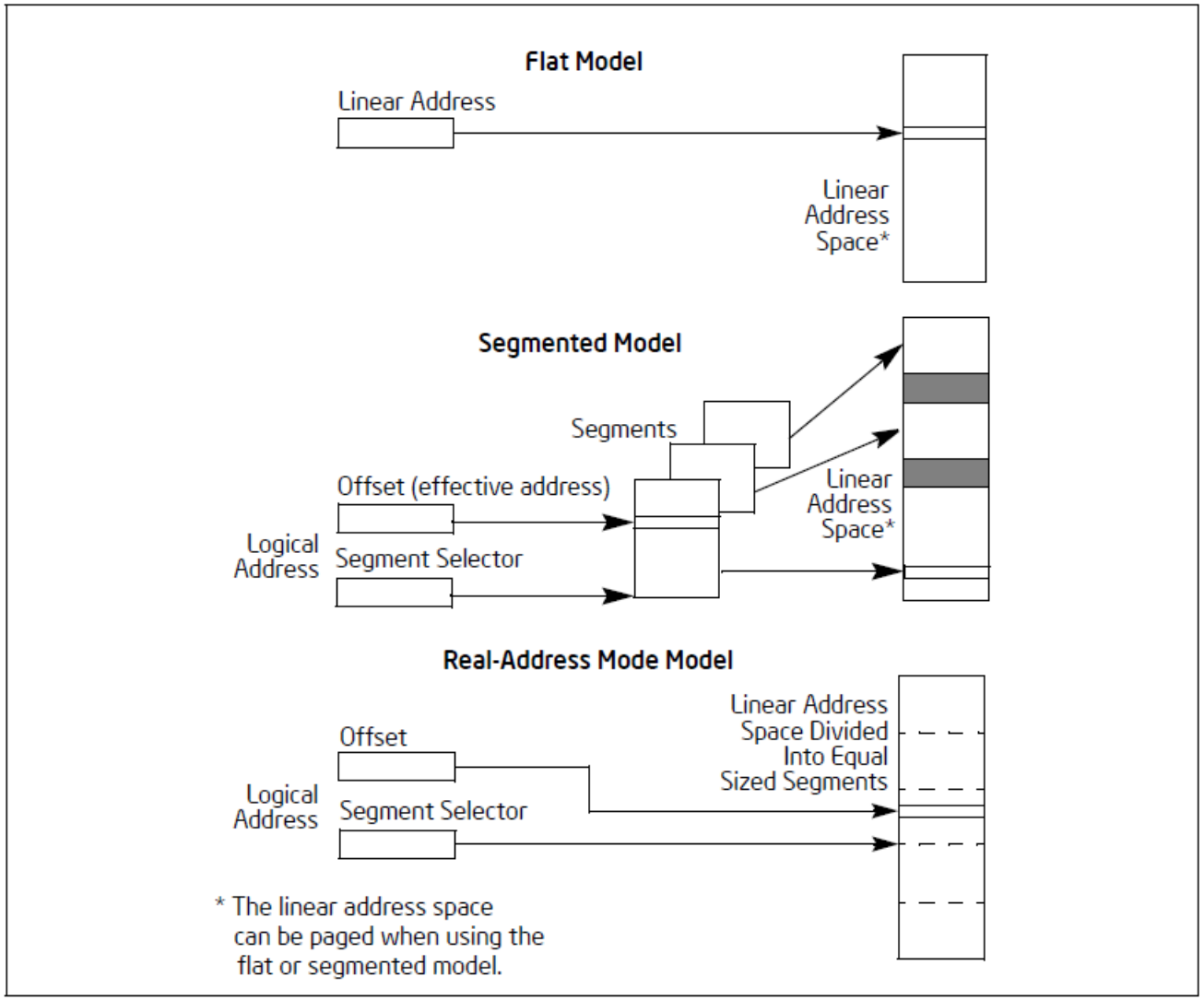


Figure 3-3. Three Memory Management Models

Referências

- **TANENBAUM, A.S. Sistemas operacionais Modernos, 3ª. ed.**
  - Cap. 3: 3.2, 3.3.1 até 3.3.3, 3.4
  - Assim como imagens do livro
- **Intel® 64 and IA-32 Architectures Software Developer's Manual Combined**
  - Seção 3.3

Exercícios

- **TANENBAUM, A.S. Sistemas operacionais Modernos, 3ª. ed.**
  - Cap. 3: 37, 38