MC504 - Sistemas Operacionais

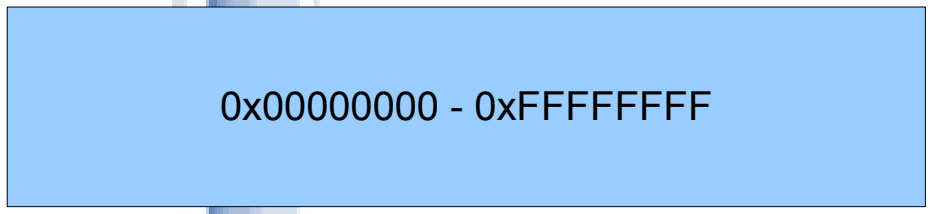
Gerência de Memória II

Islene Calciolari Garcia
Instituto de Computação - Unicamp
Primeiro Semestre de 2017

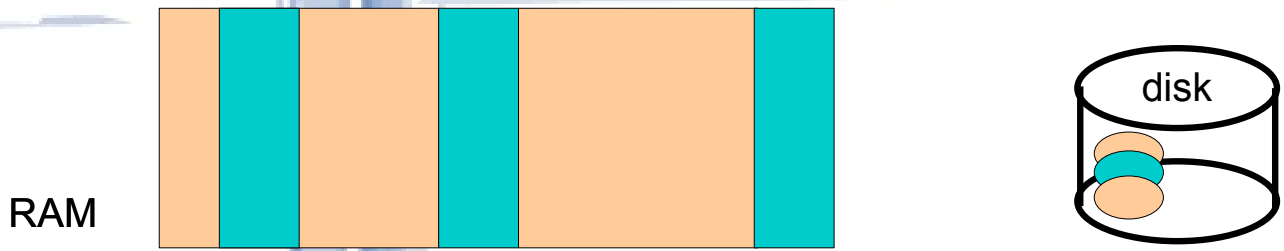
Sumário

What is Virtual Memory?

- A technique that gives each process the illusion of a private, contiguous address space



- From fragments of physical RAM and disk



Fonte: Troy D. Hanson

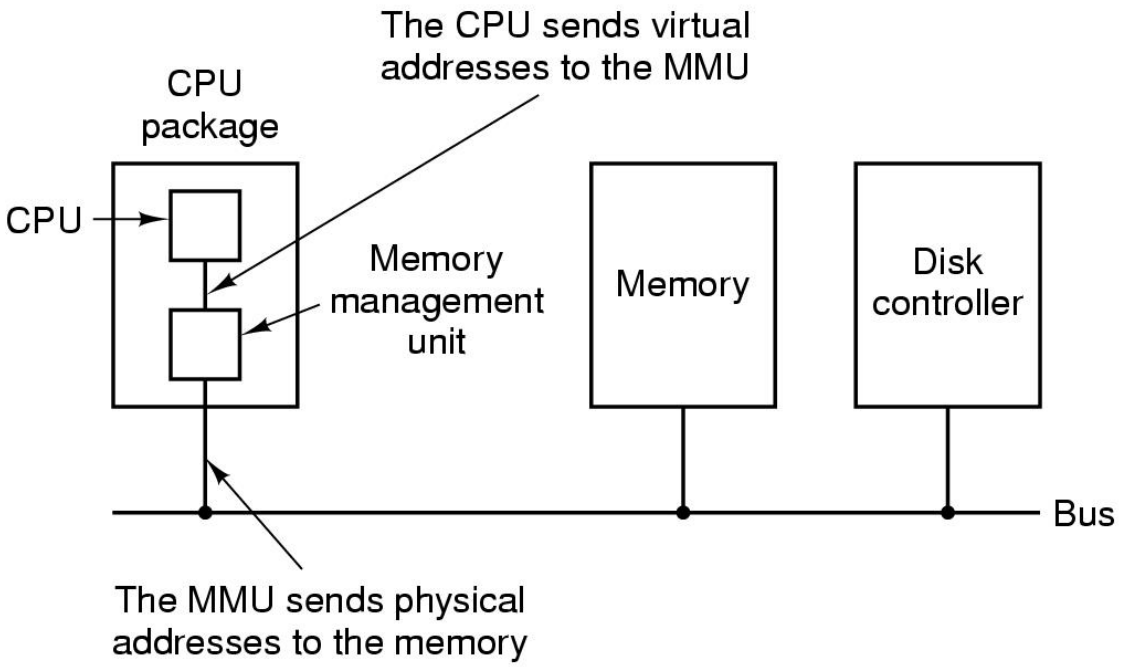
virtual → physical

- Software deals with **virtual** addresses
- Hardware needs **physical** addresses

0x12345678 → ???

Fonte: Troy D. Hanson

Memória Virtual

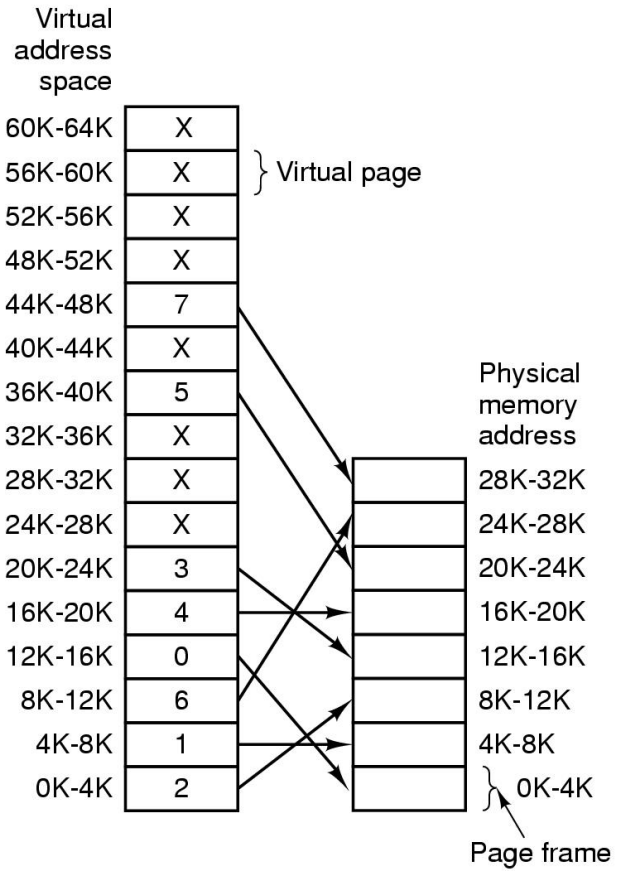


Memory Management Unit (MMU)

Tanenbaum: Figura 4.9

Memória virtual maior do que a memória física

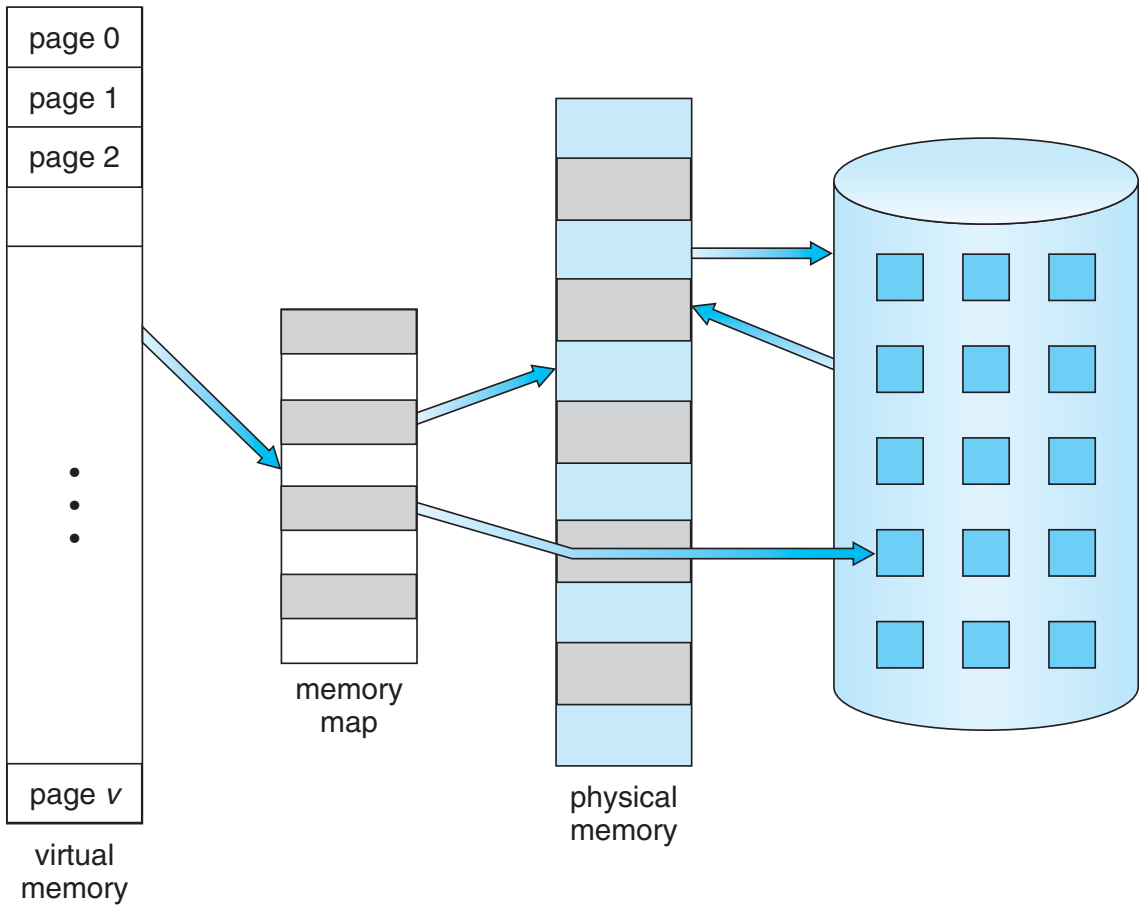
Paginação



Tanenbaum: Figura 4.10

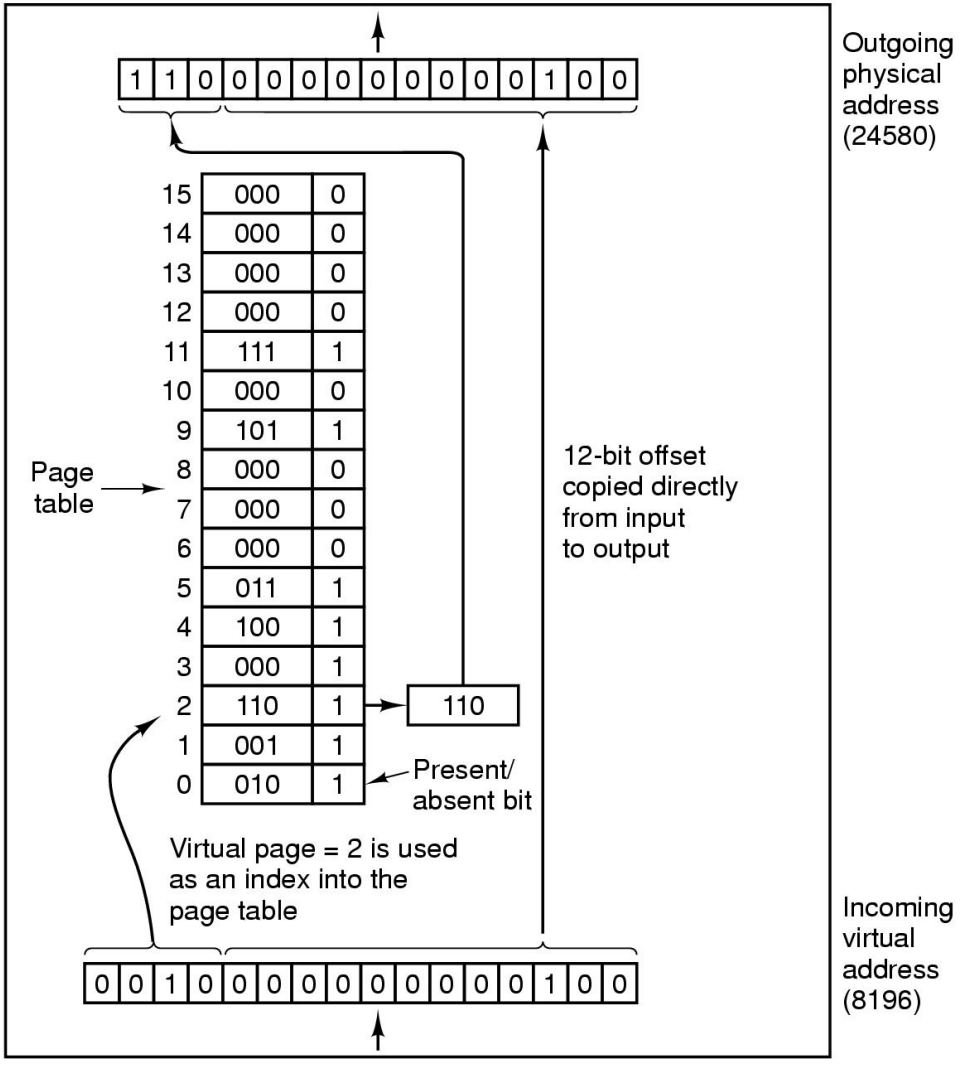
Memória virtual maior do que a memória física

Paginação e armazenamento no disco



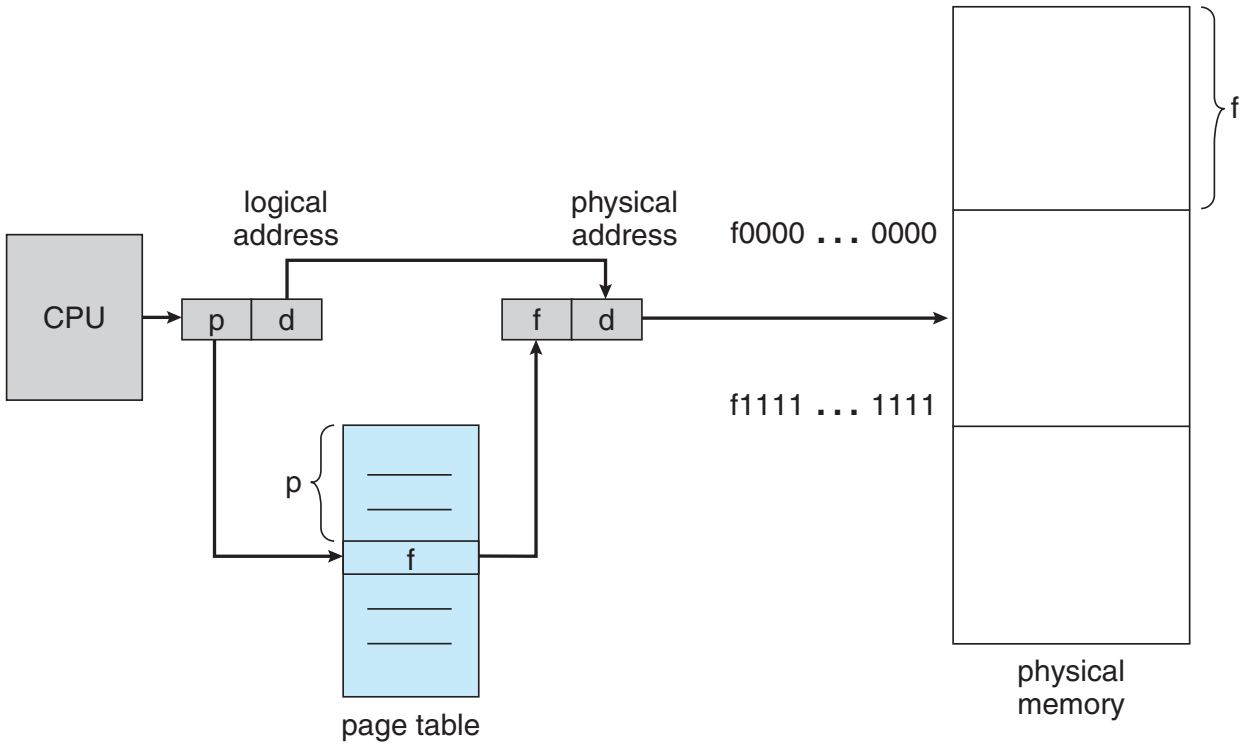
Silberschatz: Figura 9.01

Mapeamento básico de endereços



Tanenbaum: Figura 4.11

Mapeamento básico de endereços



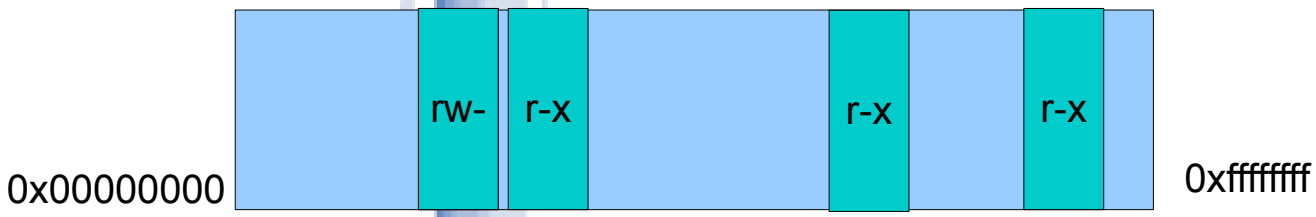
Silberschatz: Figura 8.10

Paginação - Exemplo

- 32 bits de endereço
- páginas de 4k
- 20 primeiros bits indicam a página
- 12 últimos bits indicam o deslocamento dentro da página

- Veja o código pagesize.c

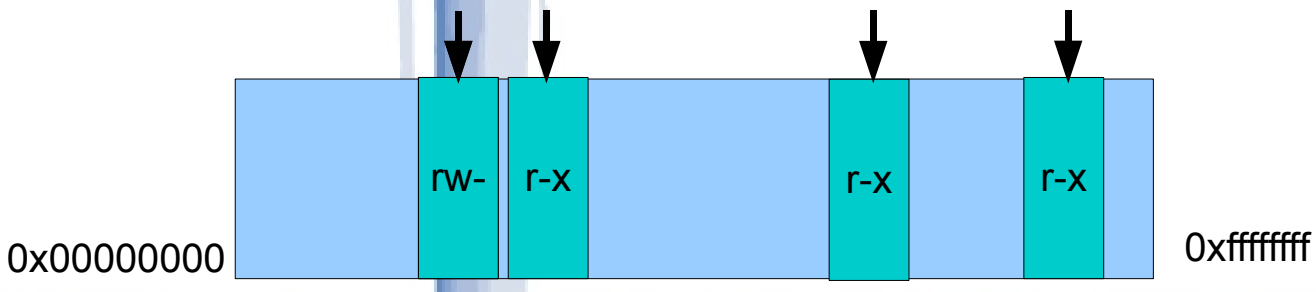
Even though the whole space is addressable,
only certain areas are legal for *r-w-x*



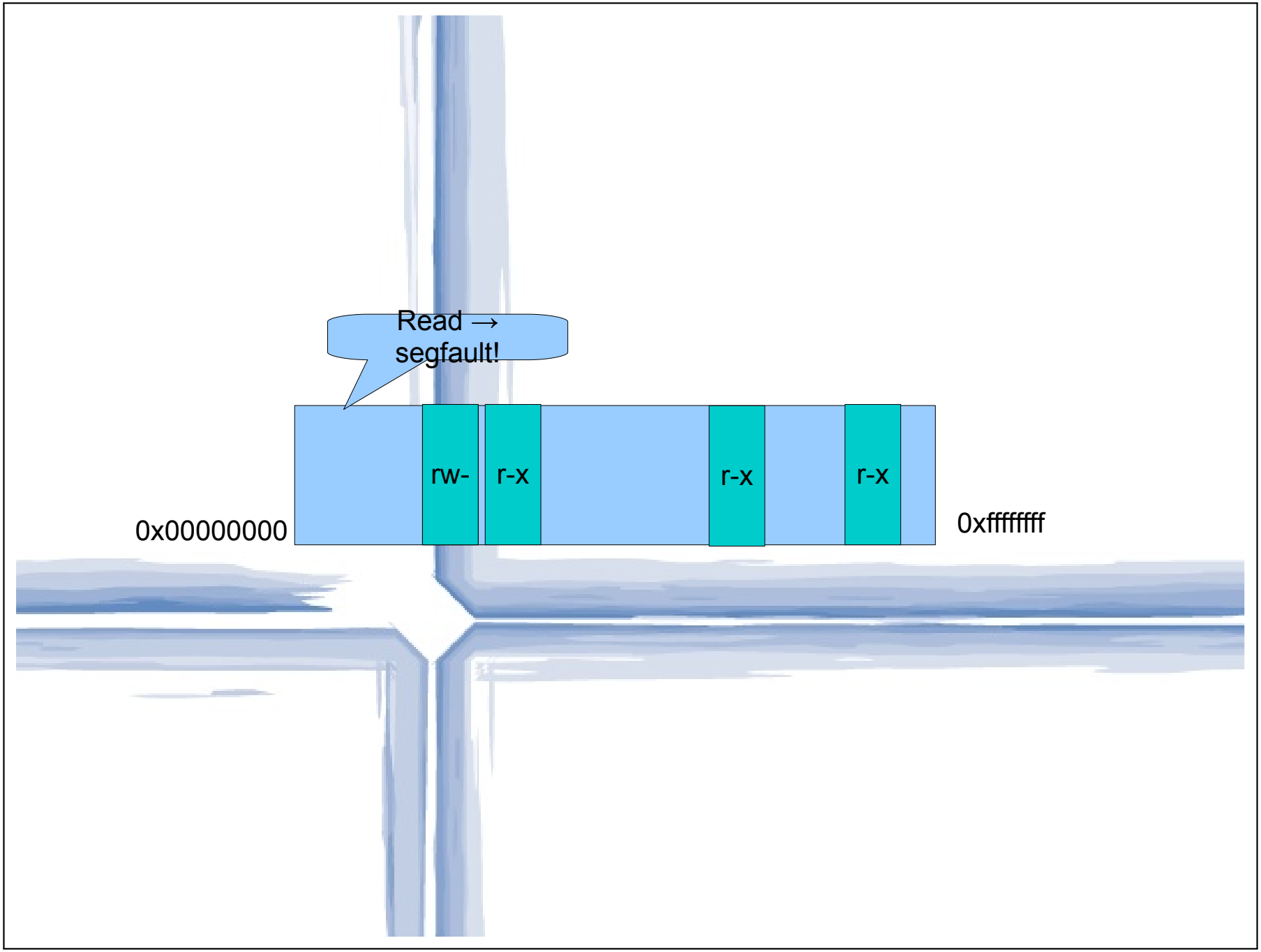
Fonte: Troy D. Hanson

These legally addressable areas are called

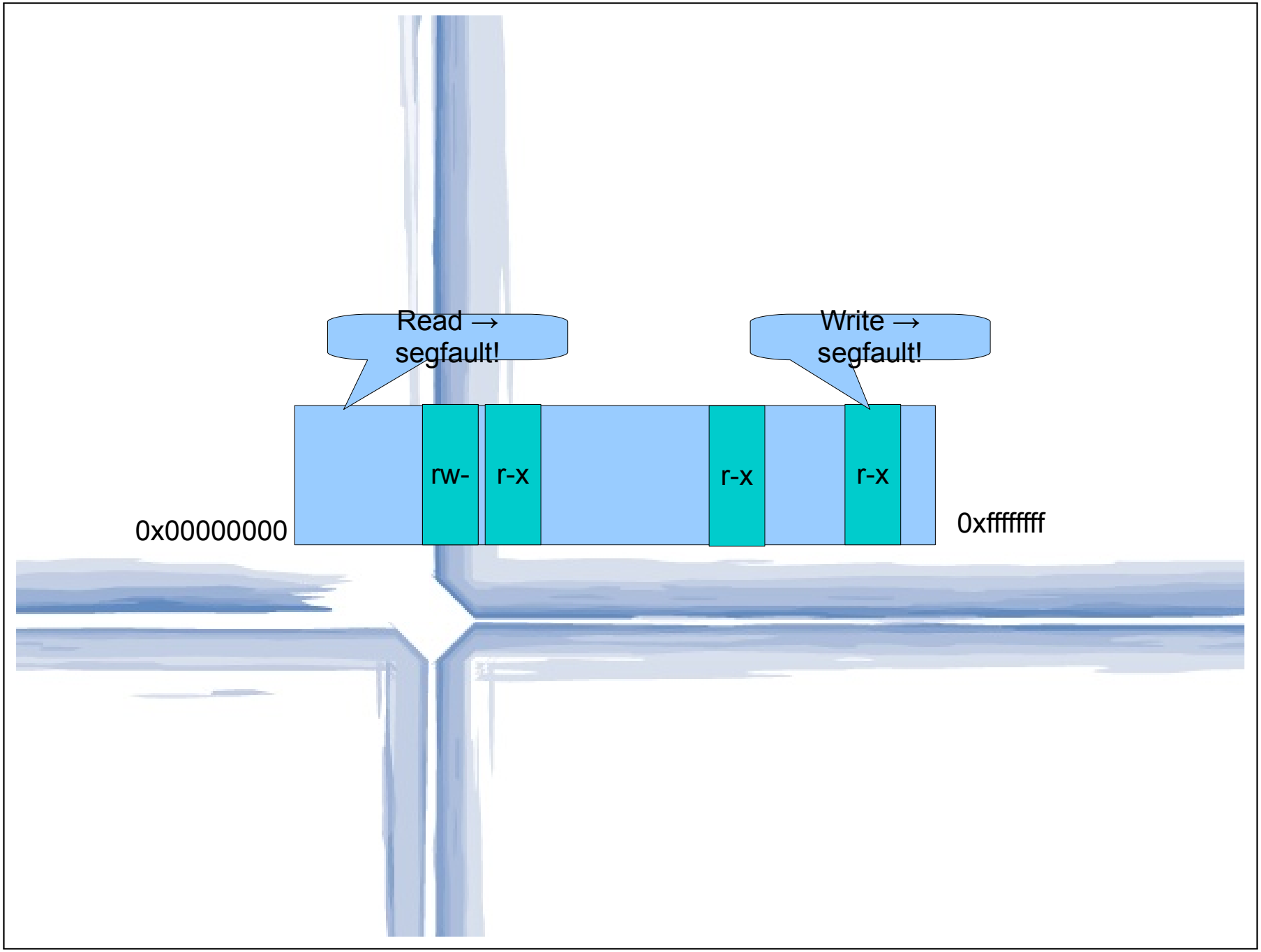
- ***Virtual Memory Areas***



Fonte: Troy D. Hanson

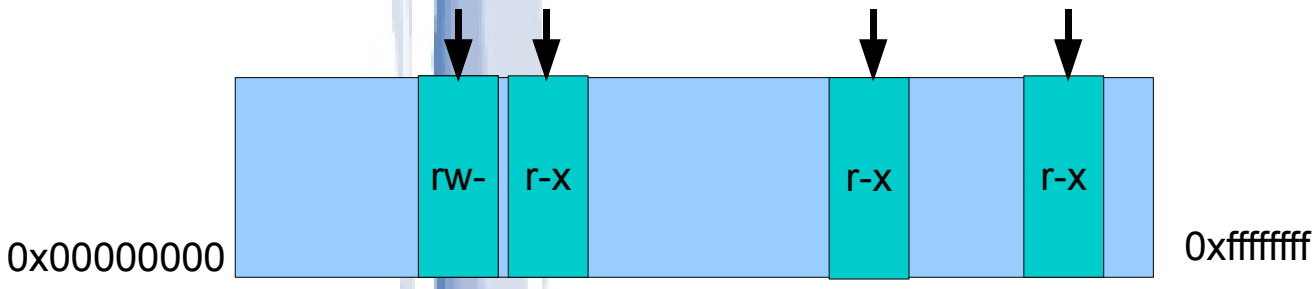


Fonte: Troy D. Hanson



Fonte: Troy D. Hanson

Virtual Memory Areas

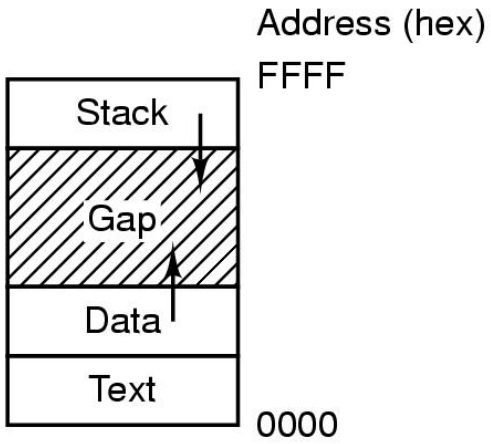



```
[thanson@linux02]$ pmap 20580
0043e000      88K r-x--  /lib/ld-2.3.4.so
00454000       4K r-x--  /lib/ld-2.3.4.so
00455000       4K rwx--  /lib/ld-2.3.4.so
00458000    1176K r-x--  /lib/tls/libc-2.3.4.so
0057e000       8K r-x--  /lib/tls/libc-2.3.4.so
00580000       8K rwx--  /lib/tls/libc-2.3.4.so
00582000       8K rwx--    [ anon ]
08048000       4K r-x--  /home/thanson/trash/csleep
08049000       4K rw---  /home/thanson/trash/csleep
b7f9d000       4K rw---    [ anon ]
bffd1000     188K rw---    [ stack ]
ffffe000       4K -----    [ anon ]
 total      1500K
```

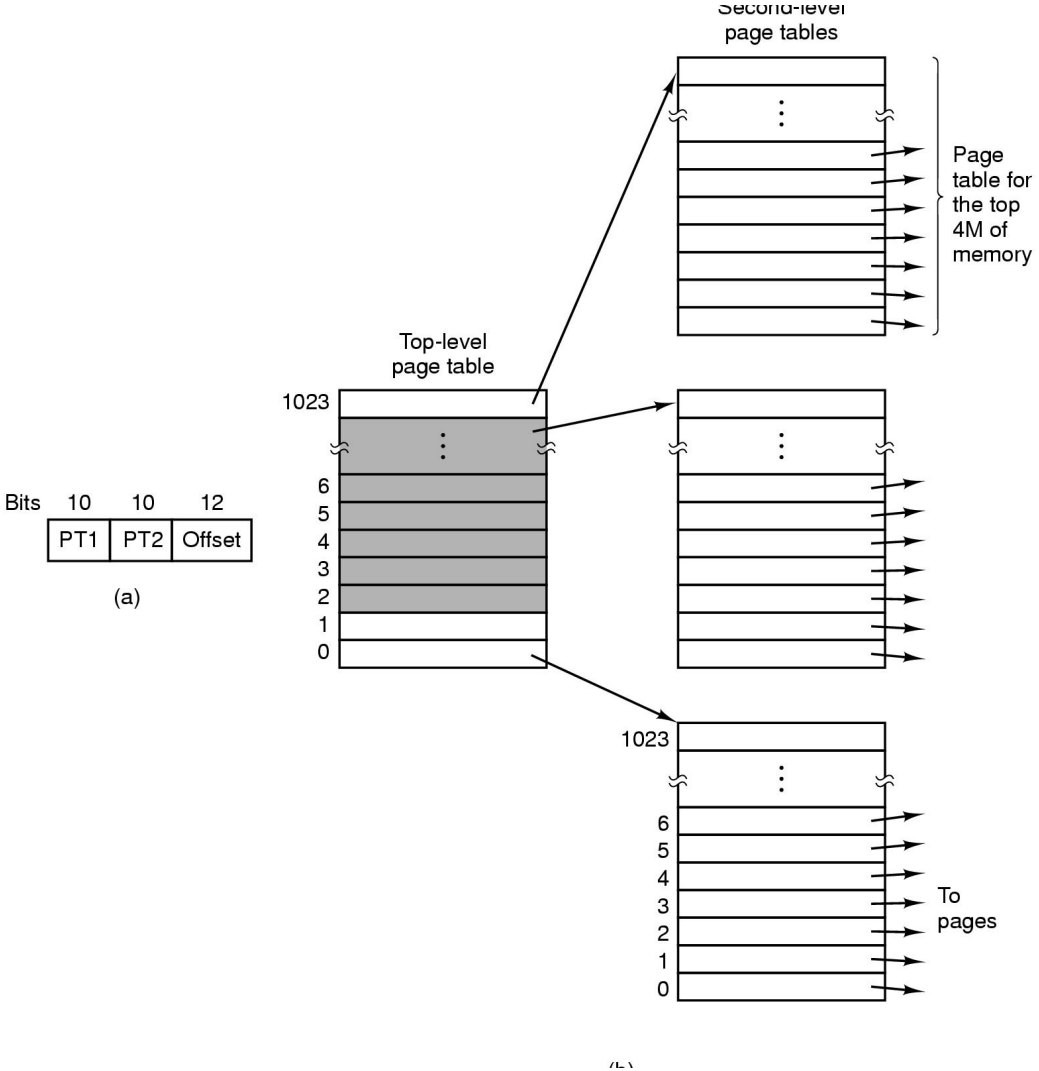
Fonte: Troy D. Hanson

Espaço de endereçamento

- Apenas as páginas ocupadas precisam ser mapeadas
- Veja o código sbrk.c

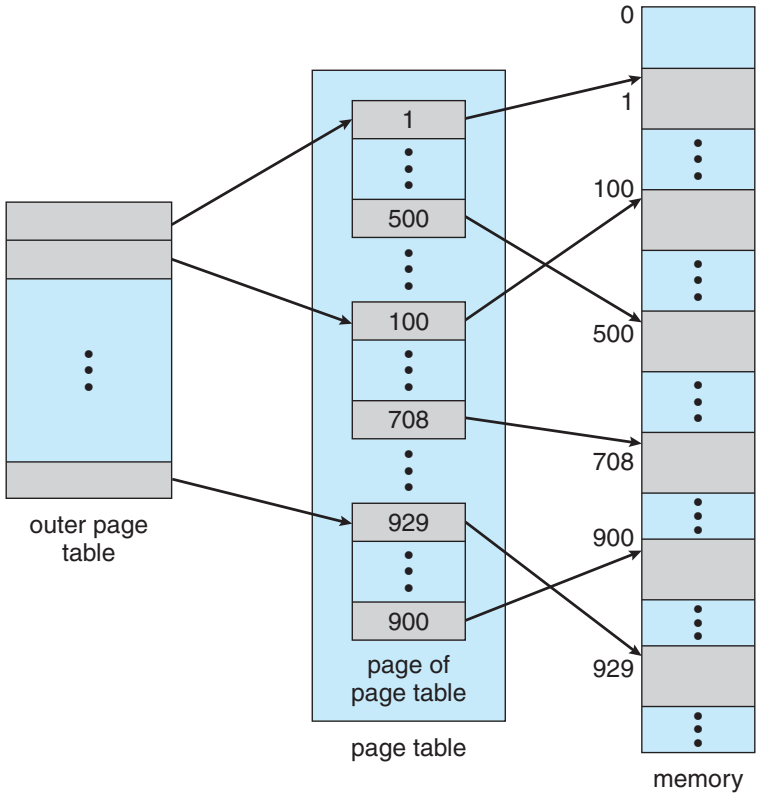


Tabelas de mais de um nível



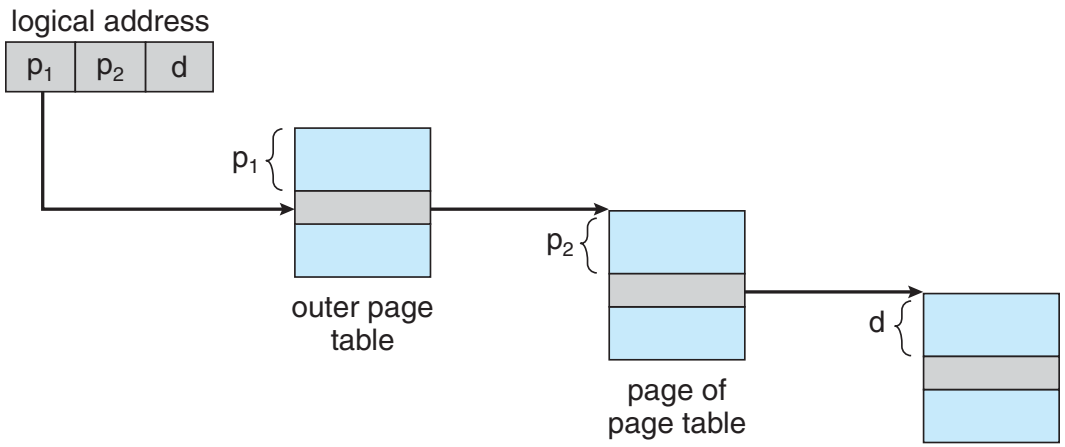
Tanenbaum: Figura 4.12

Tabelas de mais de um nível



Silberschatz: Figura 8.17

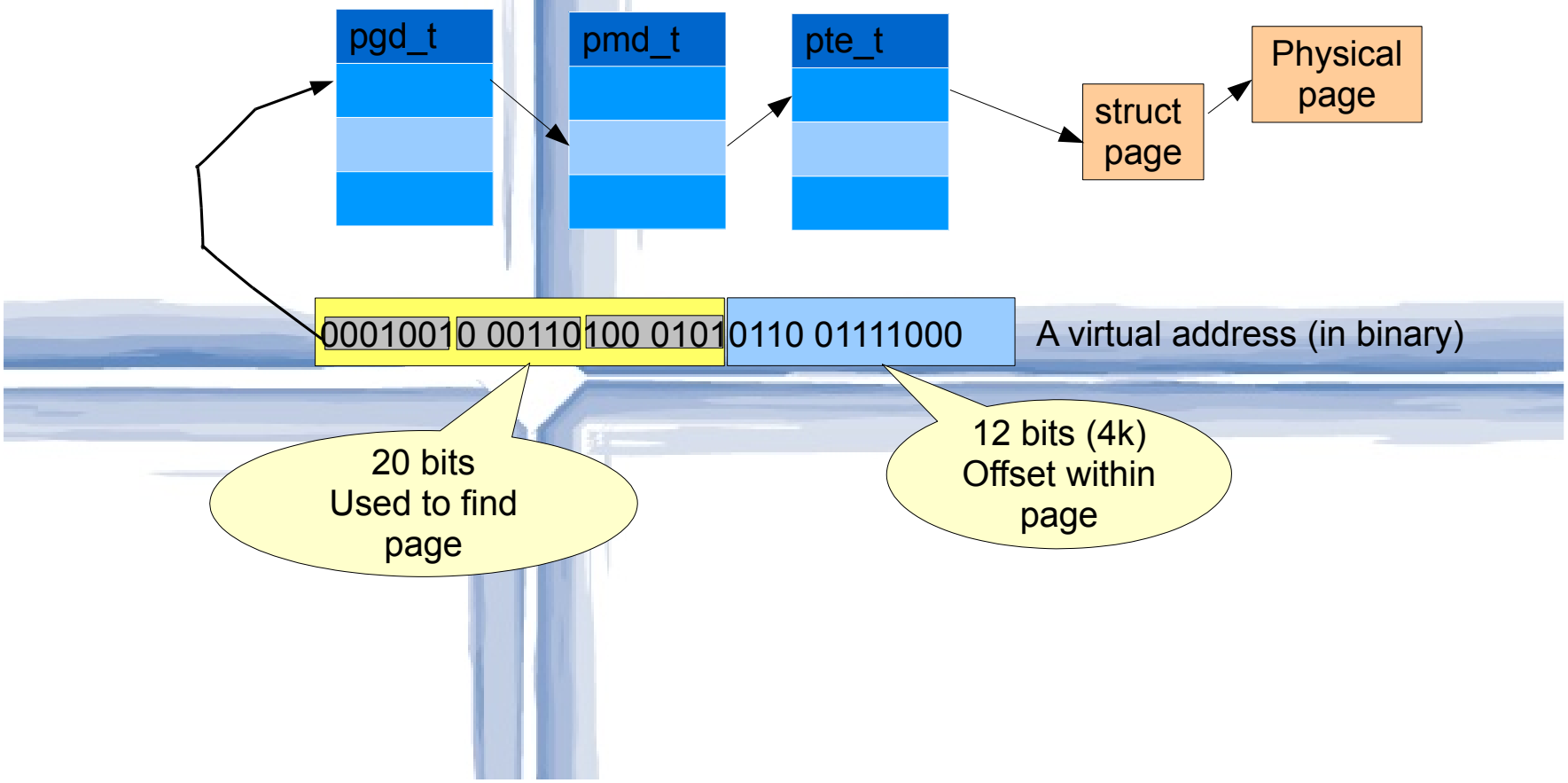
Mapeamento de endereços: tabela de dois níveis



Silberschatz: Figura 8.18

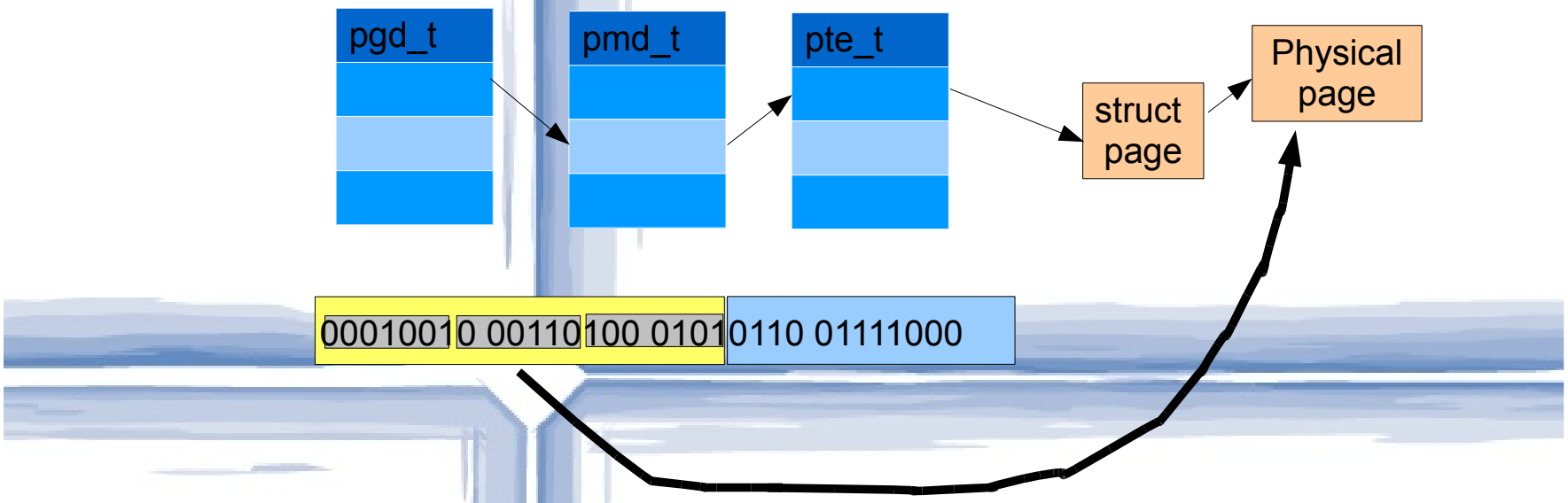
Page tables

- Linux uses a **3-level** page table scheme



Fonte: Troy D. Hanson

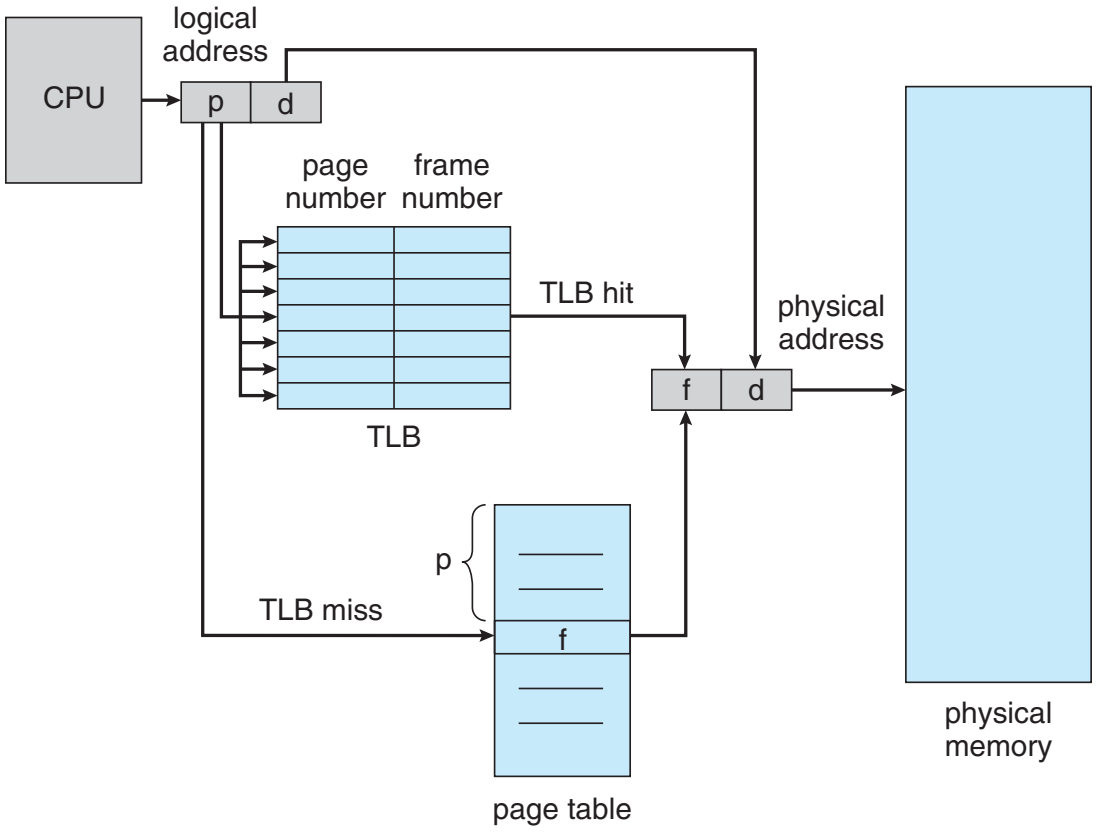
Page tables



The hardware has a small cache called the **TLB**
Translation Lookaside Buffer

Fonte: Troy D. Hanson

TLB: Translation Lookaside Buffer



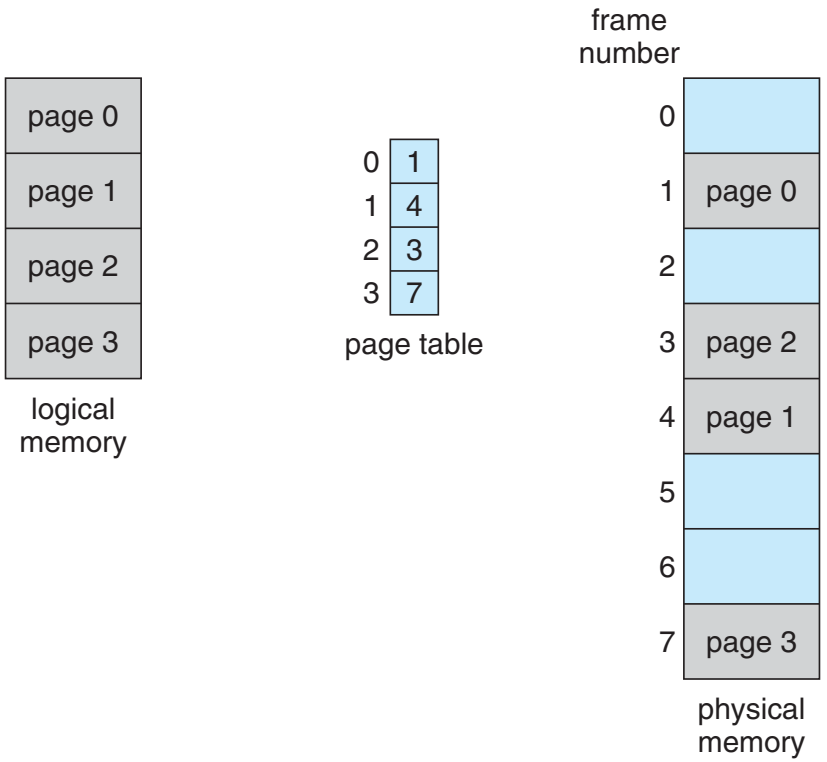
Silberschatz: Figura 8.14

TLB: Translation Lookaside Buffer

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

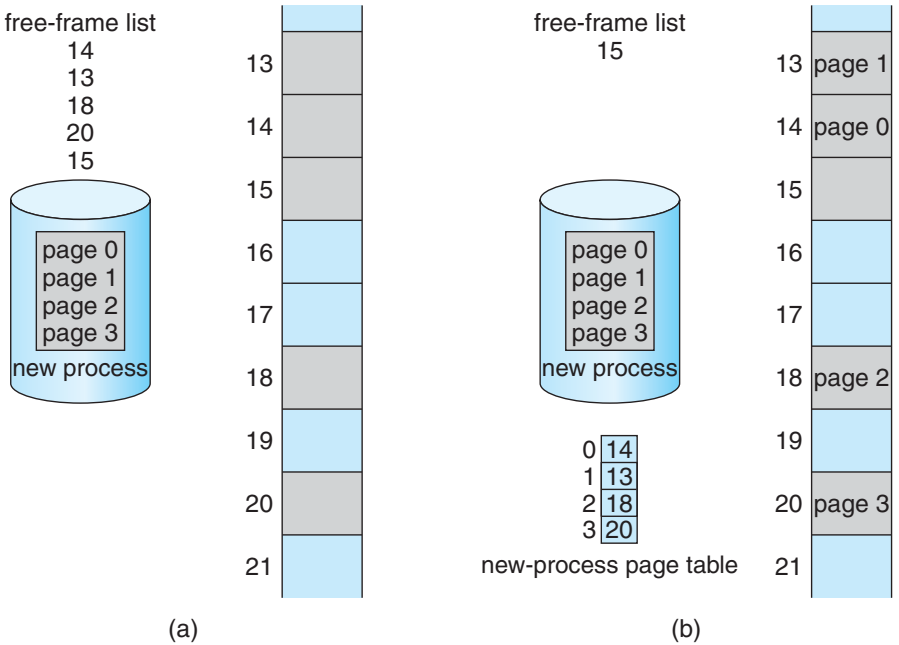
Tanenbaum: Figura 4.14

Tabelas por processo



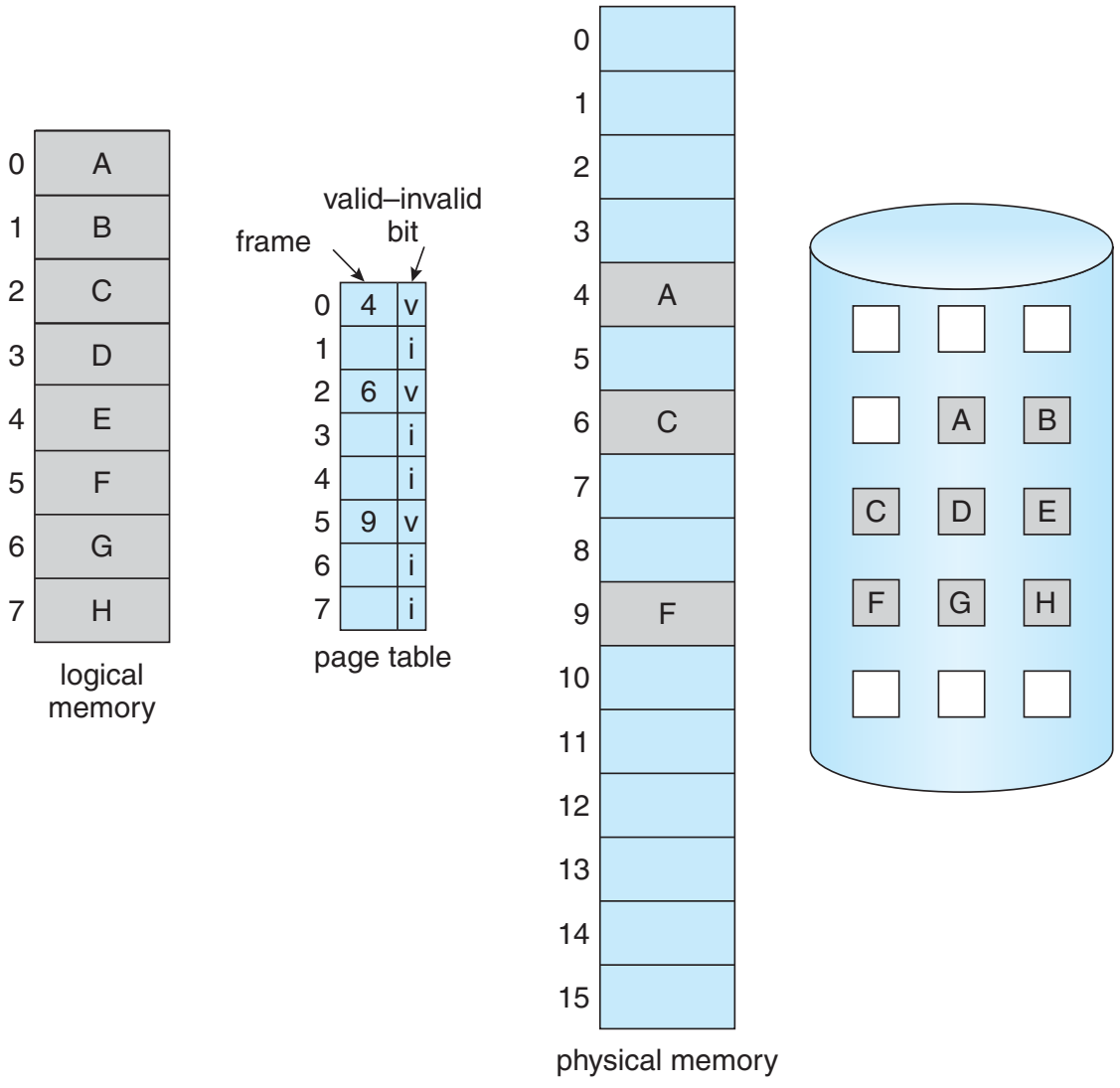
Silberschatz: Figura 8.11

Novo processo



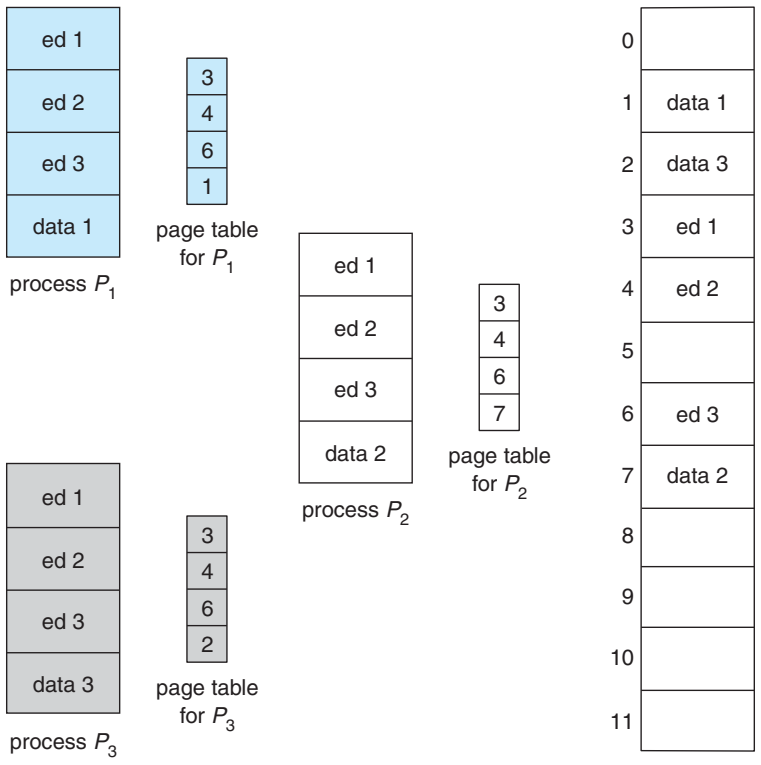
Silberschatz: Figura 8.13

Nem tudo precisa estar em memória...



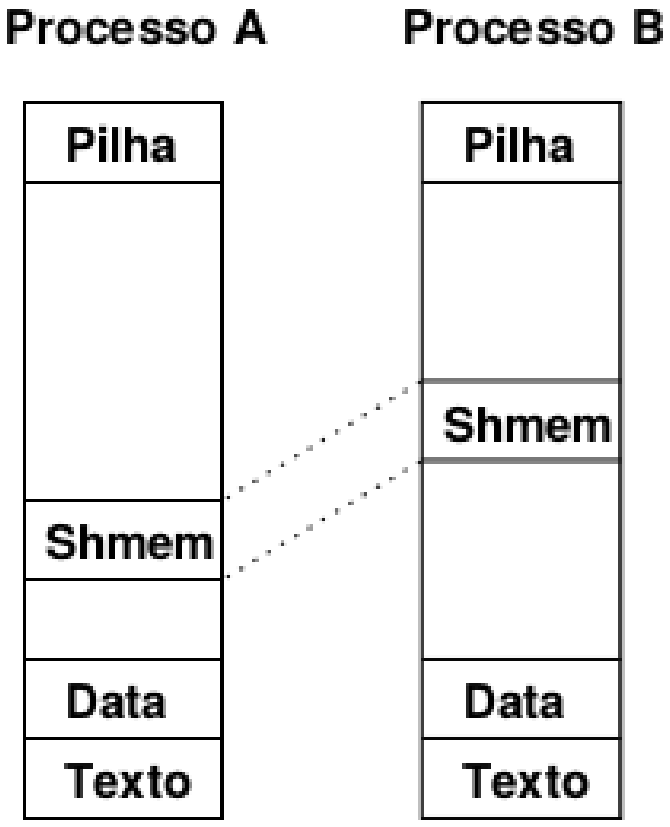
Silberschatz: Figura 9.05

Compartilhamento de áreas de código



Silberschatz: Figura 8.16

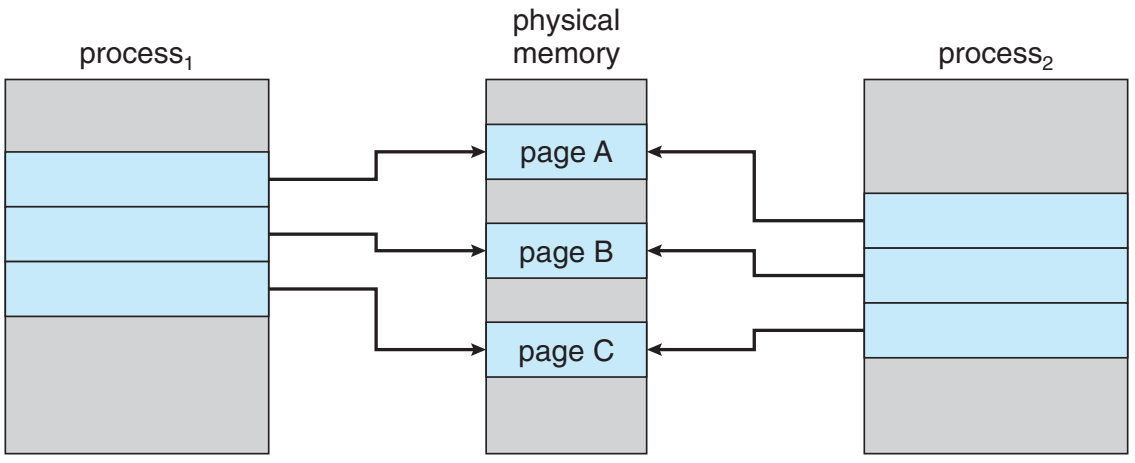
Memória compartilhada




```
int shmget(key_t key, size_t size,
           int shmflg);
void *shmat(int shmid,
            const void *shmaddr,
            int shmflg);
```

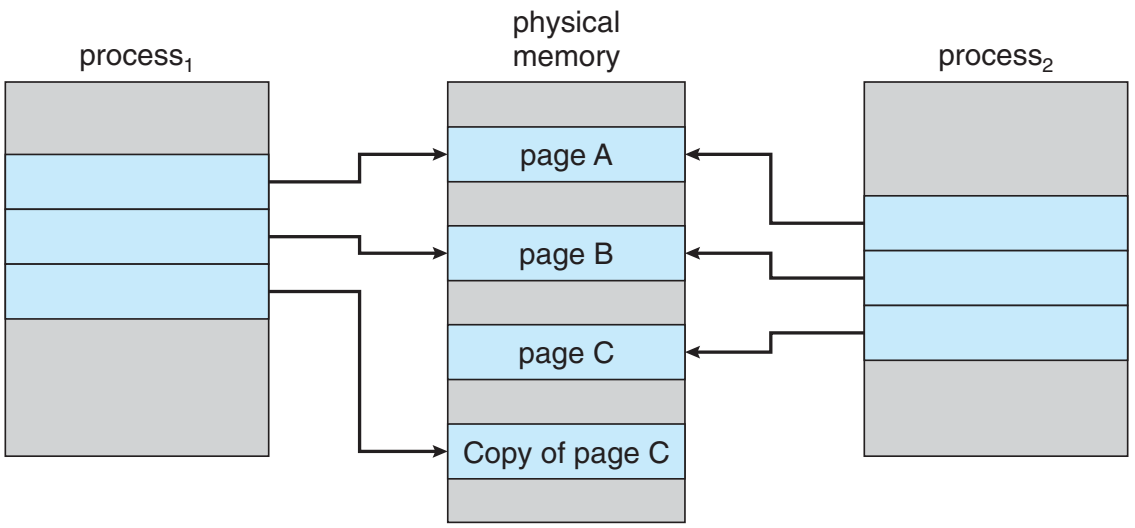
- Veja os exemplos: sh1.c sh2.c sh_fork.c sh_server.c e sh_client.c

Copy on write (antes)



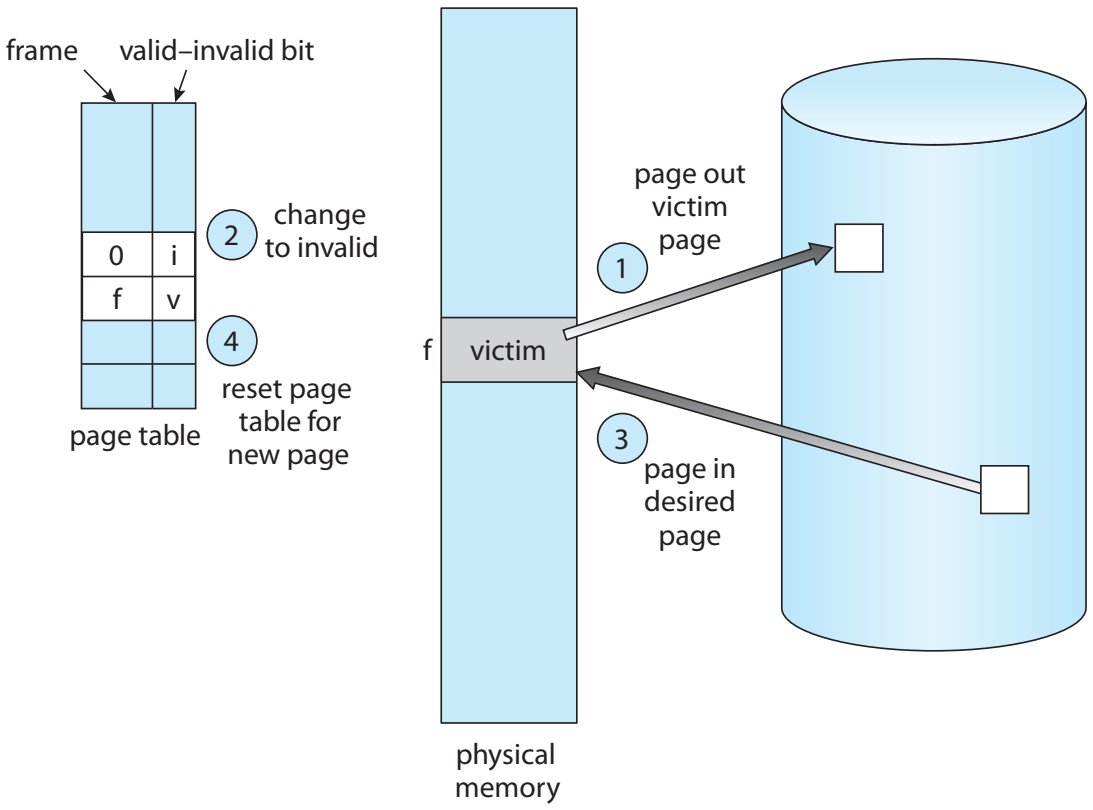
Silberschatz: Figura 9.07

Copy on write (depois)



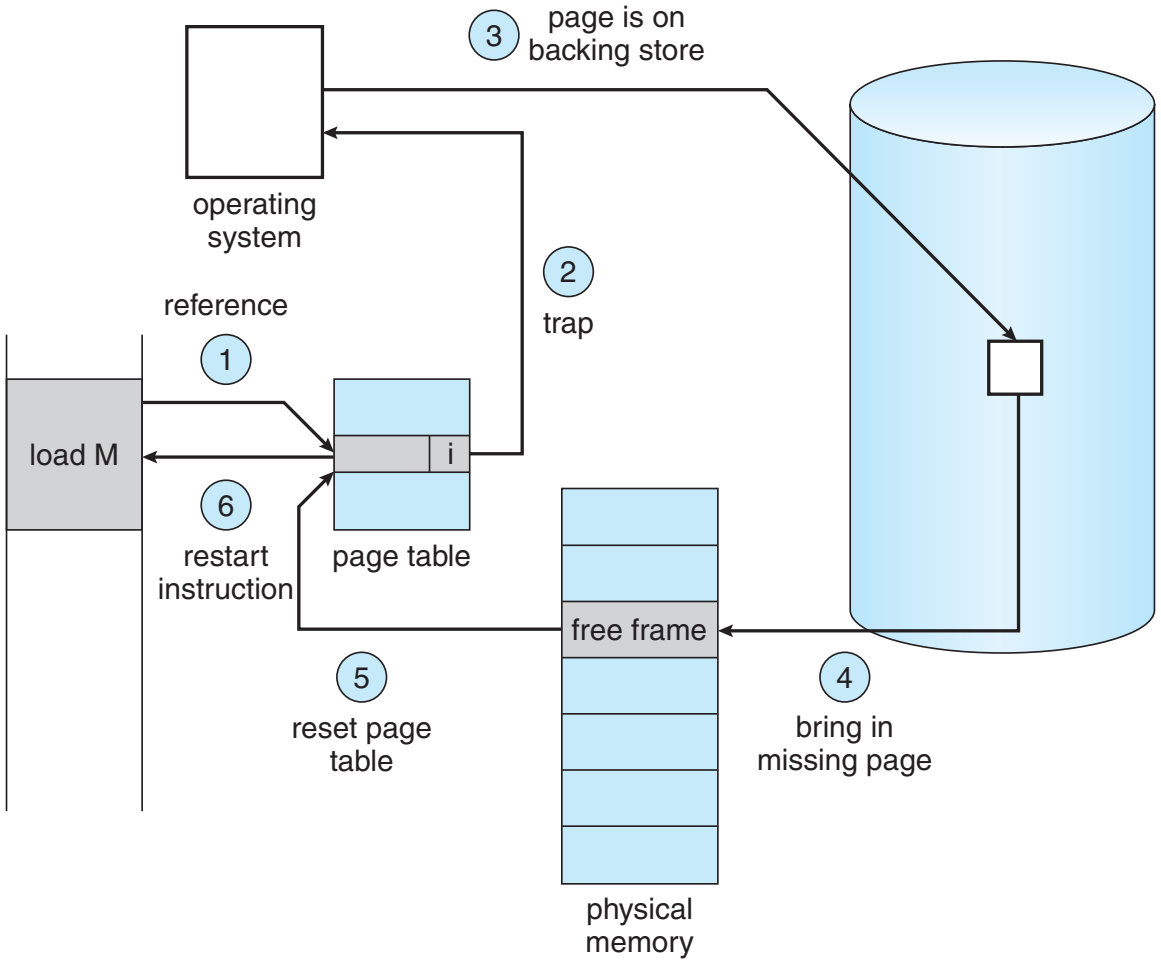
Silberschatz: Figura 9.08

Substituição de páginas



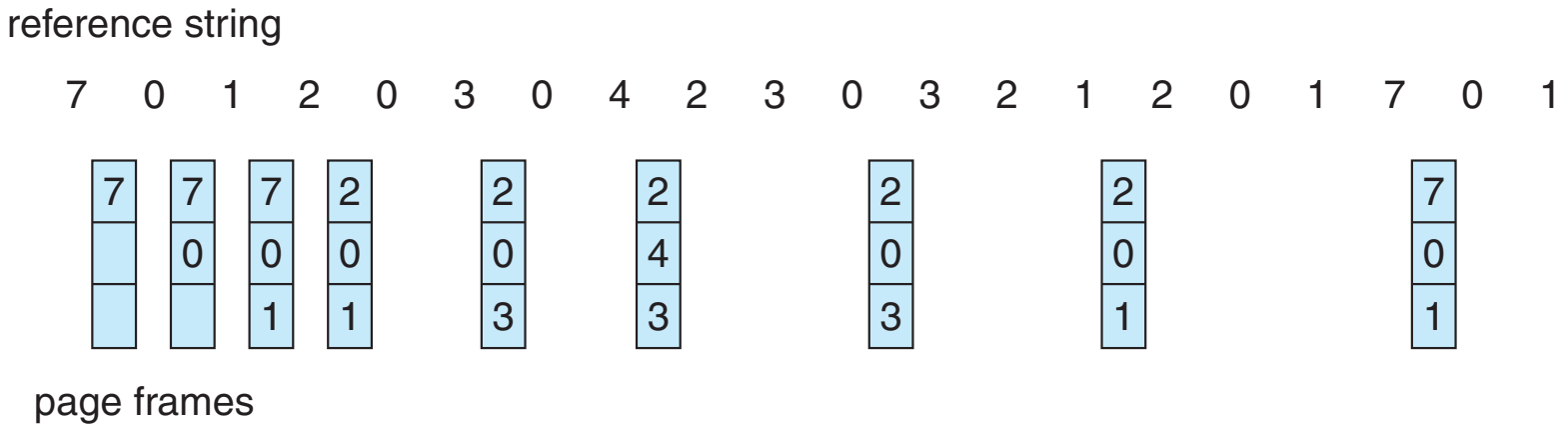
Silberschatz: Figura 9.10

Page fault



Silberschatz: Figura 9.06

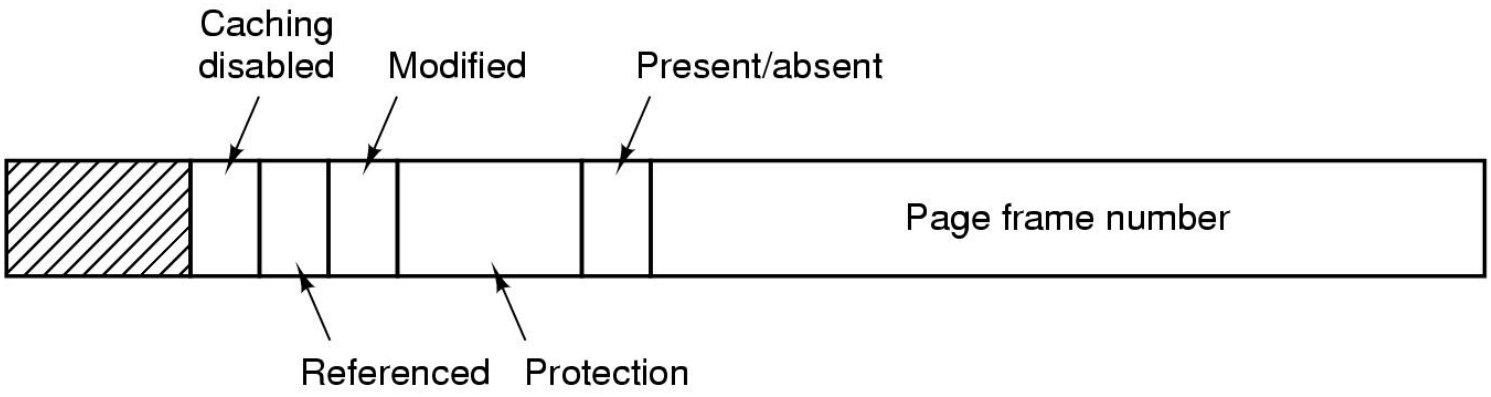
Algoritmo ótimo para substituição de páginas



Silberschatz: Figura 9.14

- Baseado no uso futuro de uma página
- Impossível de ser implementado
- Pode ser simulado (segunda execução do mesmo processo com a mesma entrada)
- Útil para comparação entre algoritmos

Entrada na tabela

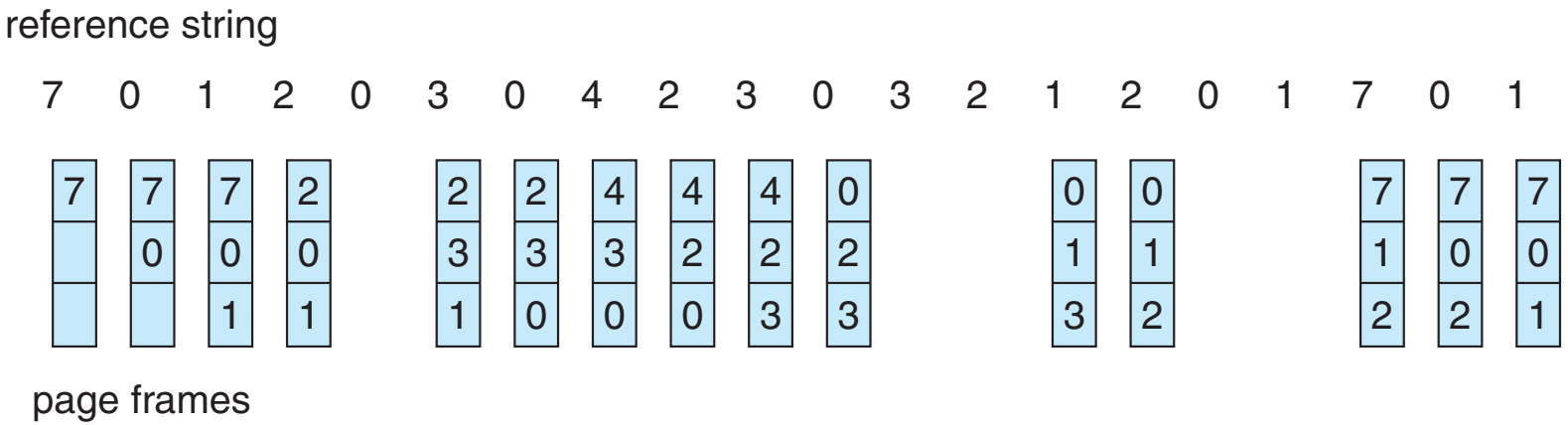


Tanenbaum: Figura 4.13

Não usada recentemente

- Classe 0: não referenciada, não modificada
- Classe 1: não referenciada, mas modificada
- Classe 2: referenciada, mas não modificada
- Classe 3: referenciada e modificada

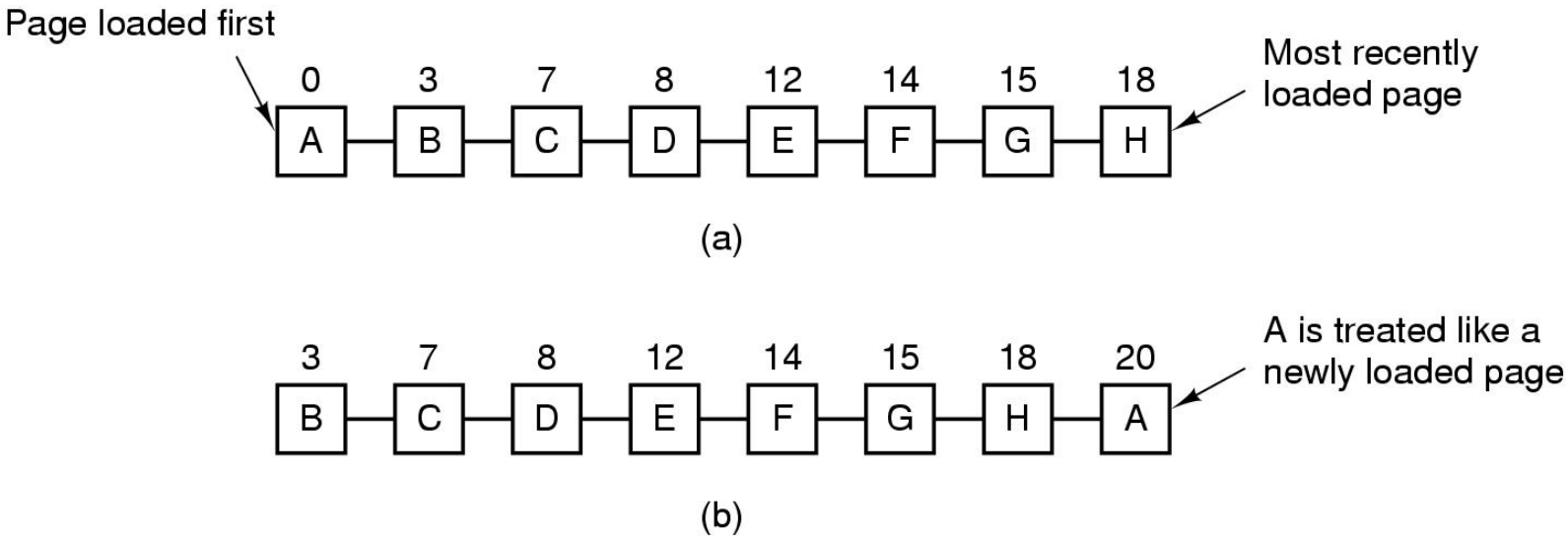
First In, First Out



Silberschatz: Figura 9.12

- Simplemente coloca as páginas em uma fila
- Pode remover páginas importantes

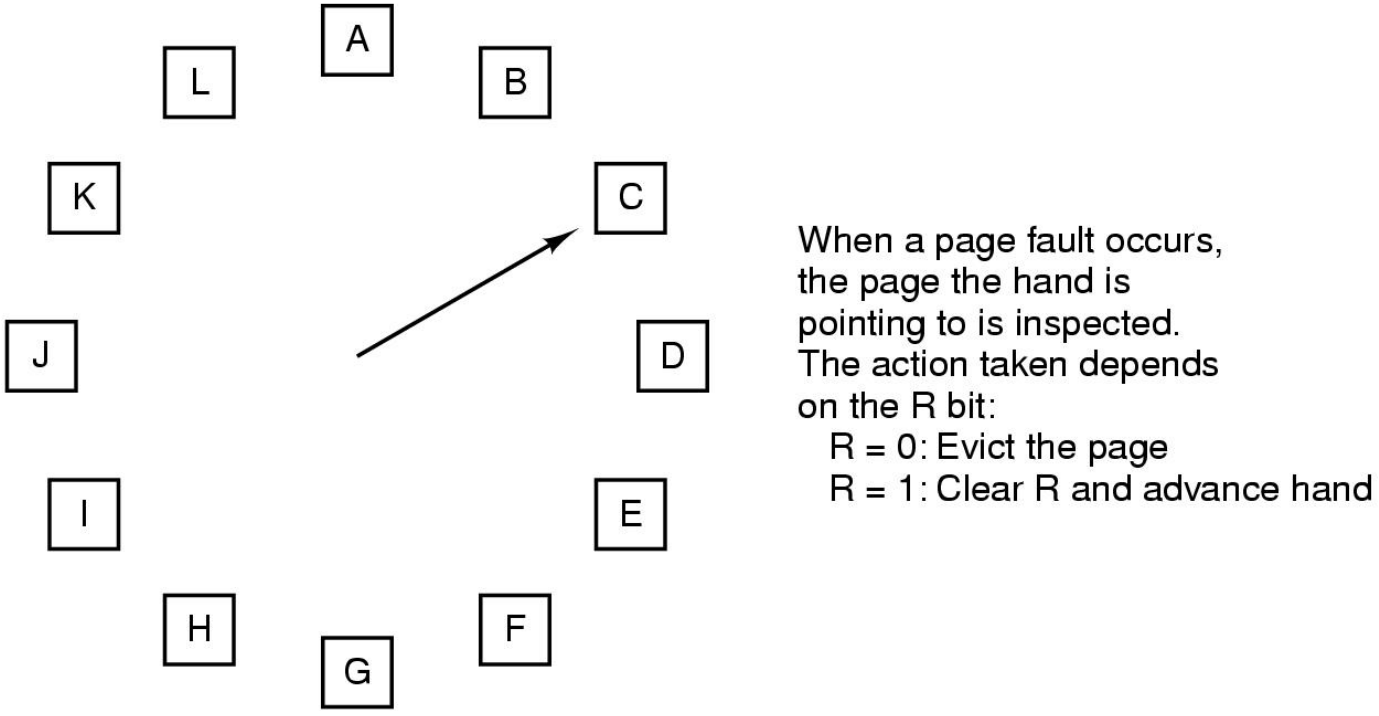
Segunda chance



Tanenbaum: Figura 4.16

- Se o bit R == 0, a página é substituída, senão
- bit R é limpo e a página é colocada no final da fila

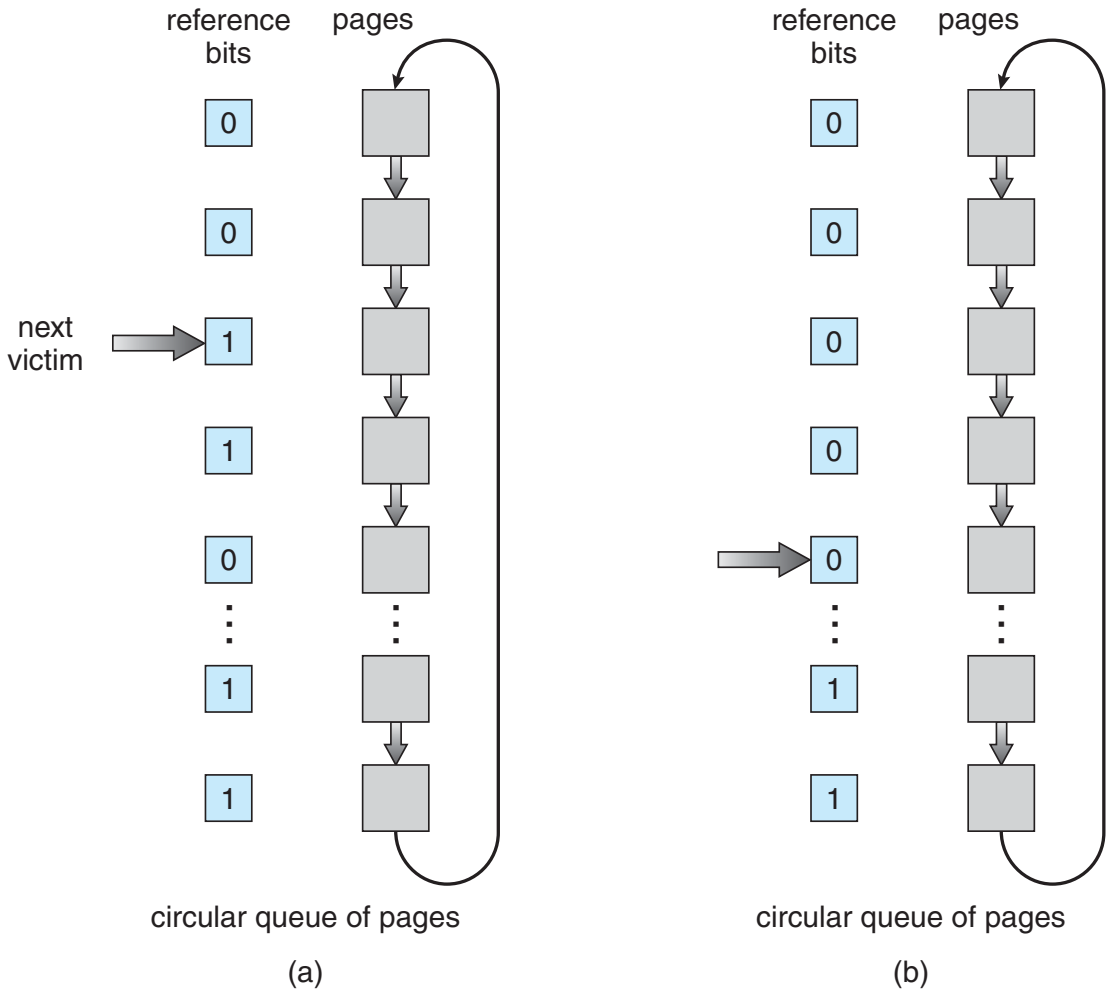
Relógio



Tanenbaum: Figura 4.17

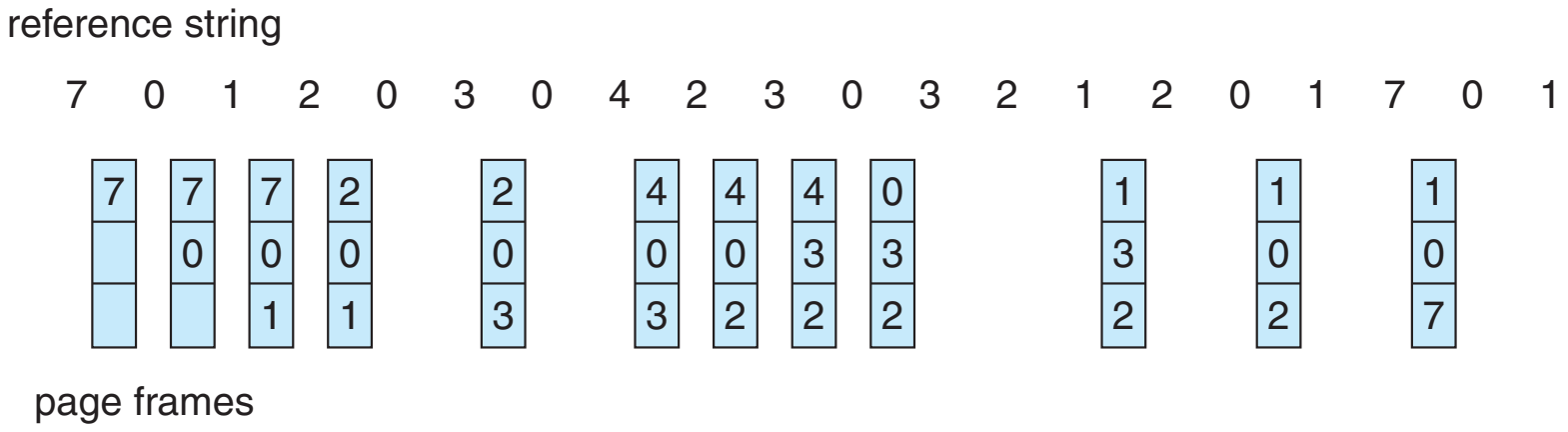
- Implementação circular da segunda chance

Segunda chance (relógio)



Silberschatz: Figura 9.17

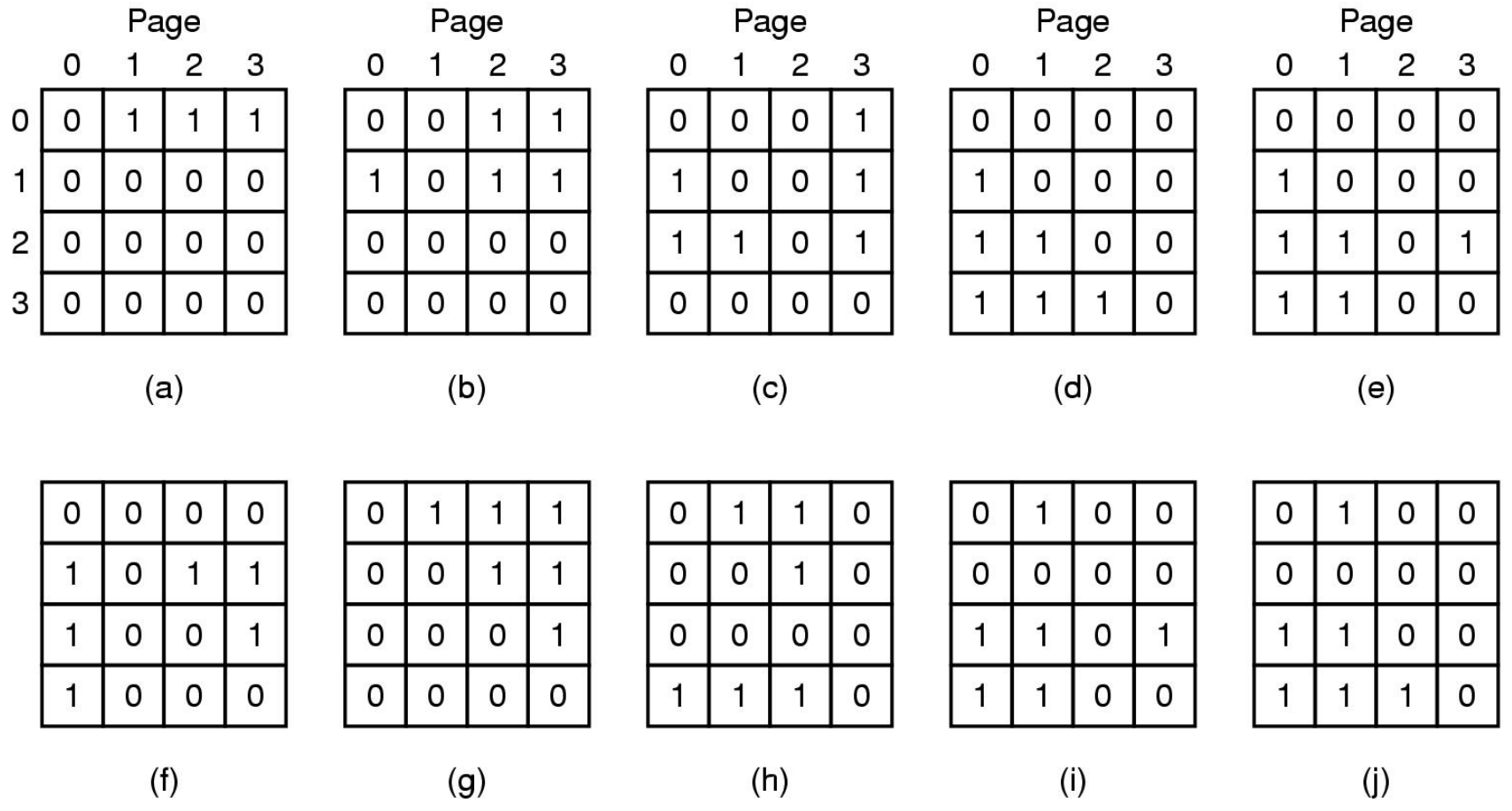
LRU (Least Recently Used)



Silberschatz: Figura 9.15

- Elimina página que não foi utilizada há mais tempo
- Implementação utilizando lista duplamente ligada
- Implementação em hardware

LRU em hardware



Acessos: 0 1 2 3 2 1 0 3 2 3

Tanenbaum: Figura 4.18

Simulando LRU em software

Página não usada frequentemente

- Contador de uso para cada página (soma o bit R a cada clock tick)
- Não esquece nada...
- Considere um compilador baseado em passos

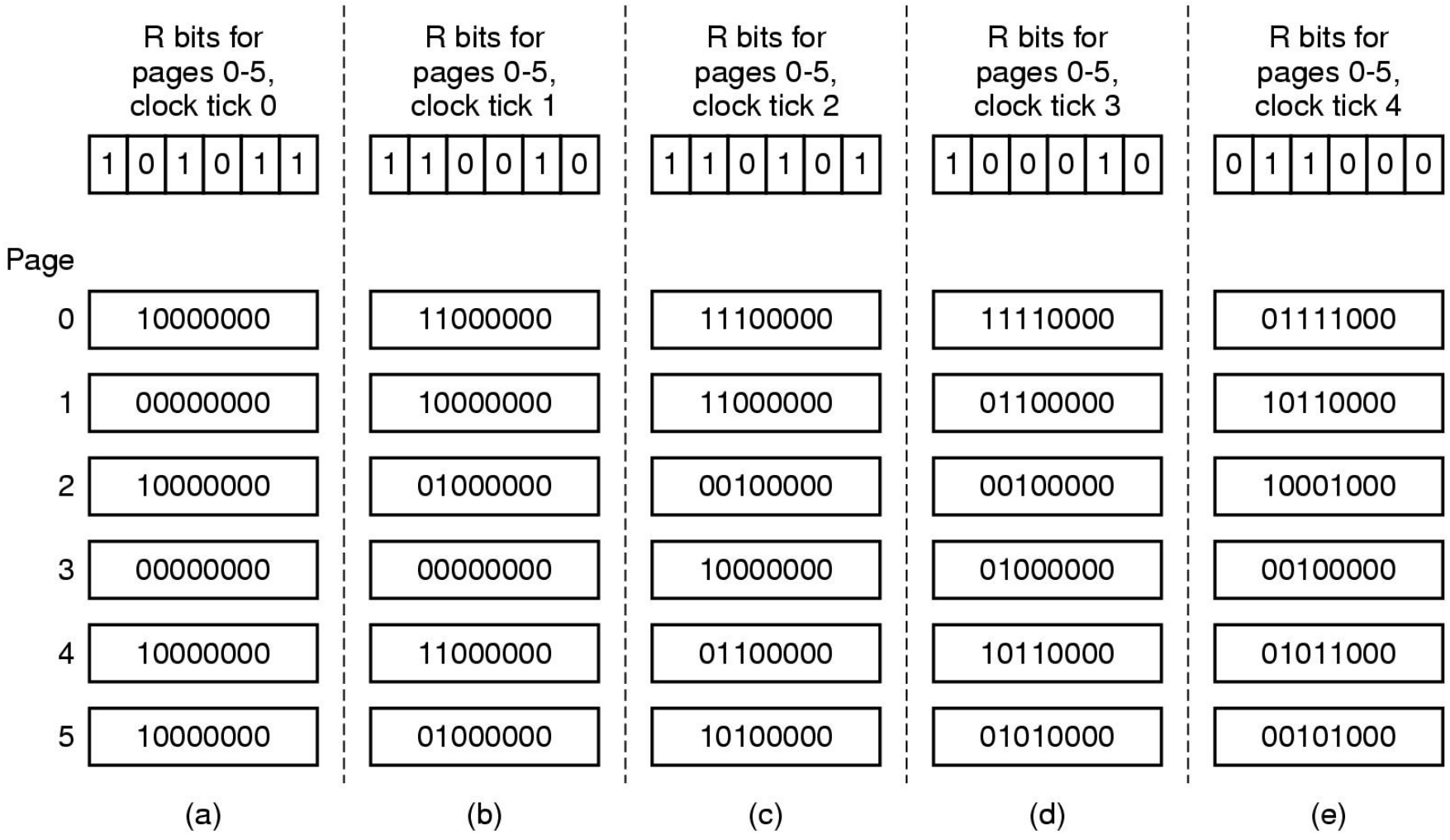
Não usada frequentemente

Aging

- O contador é deslocado à direita
- Bit R é adicionado à esquerda

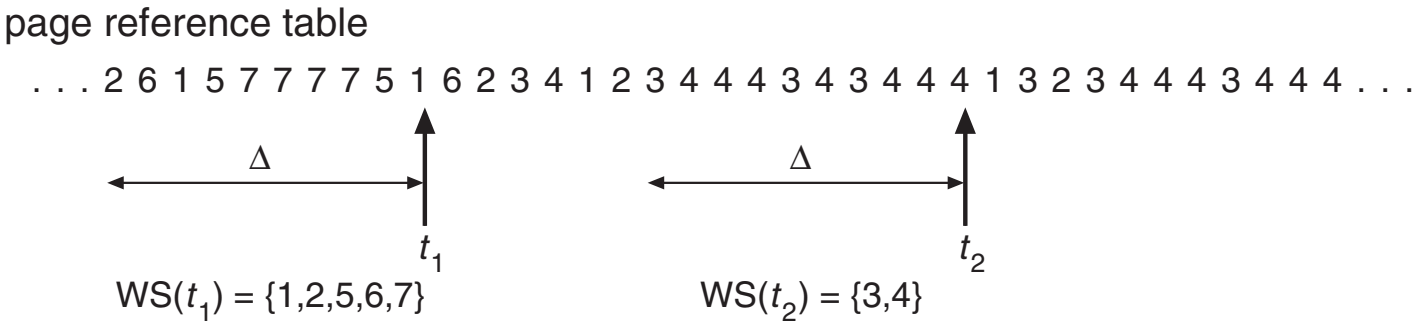
```
          1 0 1 0 1 1          1 0 1 1 0 1
>> 1        1 0 1 0 1    >> 1     1 0 1 1 0
+  1     1 1 0 1 0 1     +  0  0 1 0 1 1 0
```

Não usada frequentemente



Tanenbaum: Figura 4.19

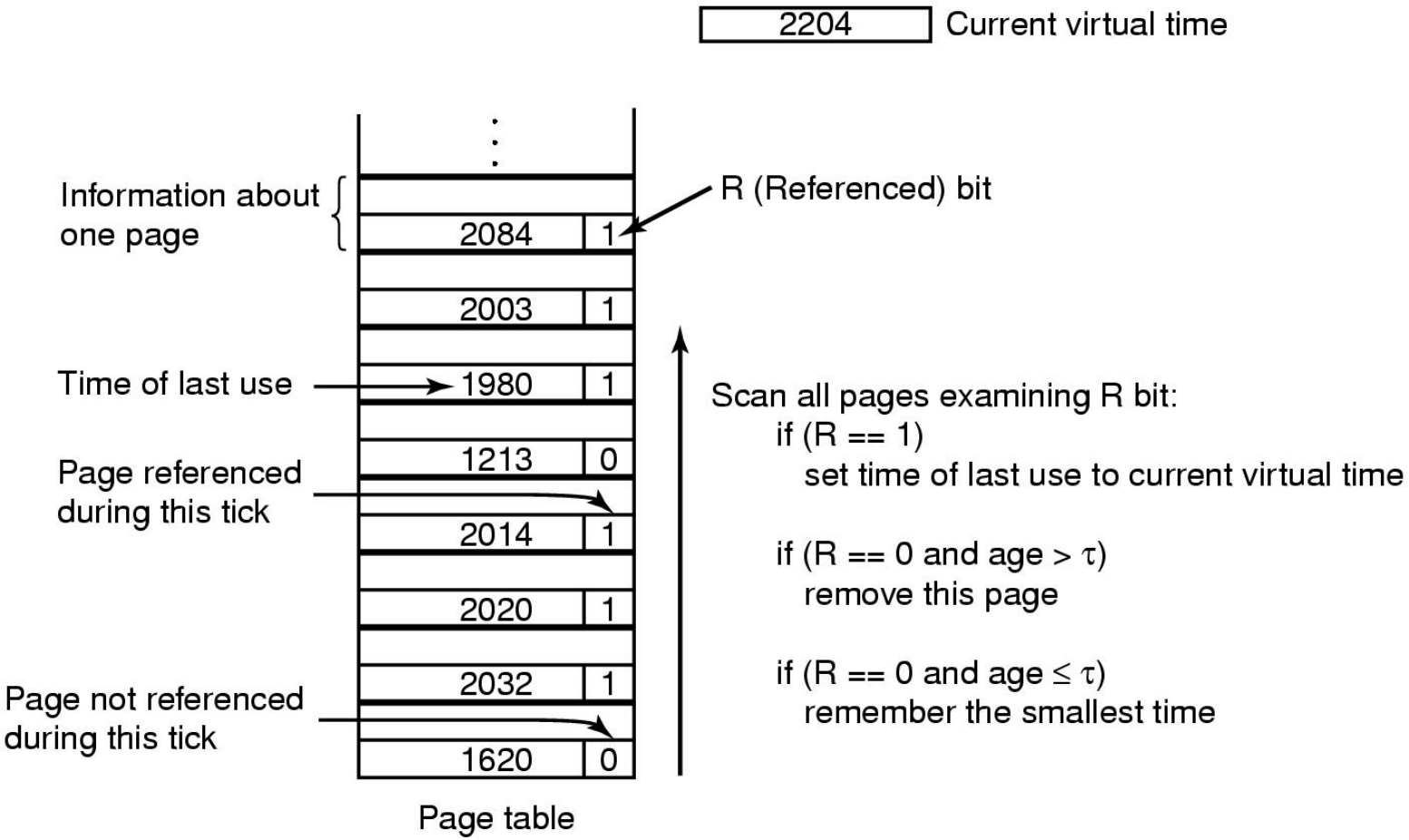
Working Set $w(k, t)$



Silberschatz: Figura 9.20

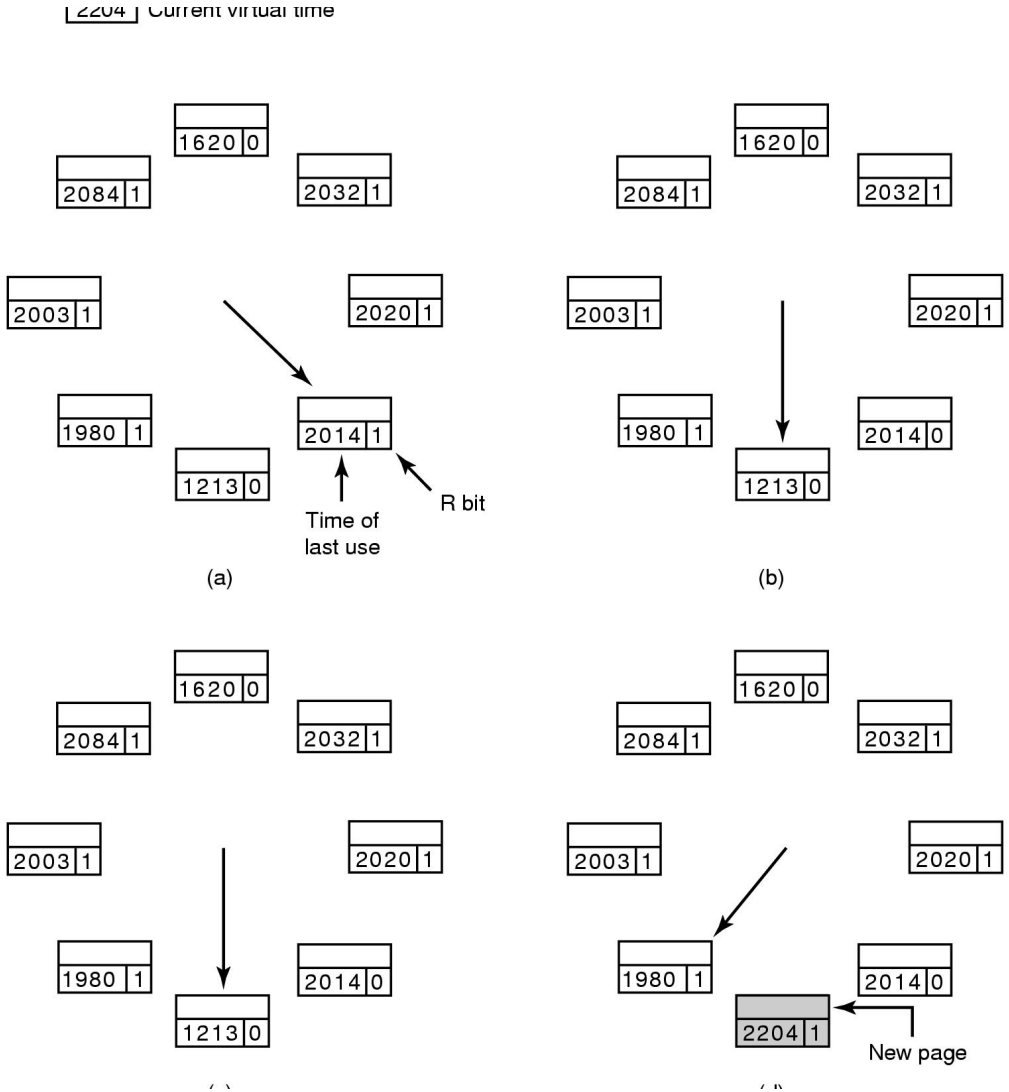
- Conjunto de páginas utilizadas nas últimas k referências em relação ao instante t.
- Paginação sob demanda
- Prepaging
- Implementação exata é muito cara

Substituição baseada em Working Set



Tanenbaum: Figura 4.21

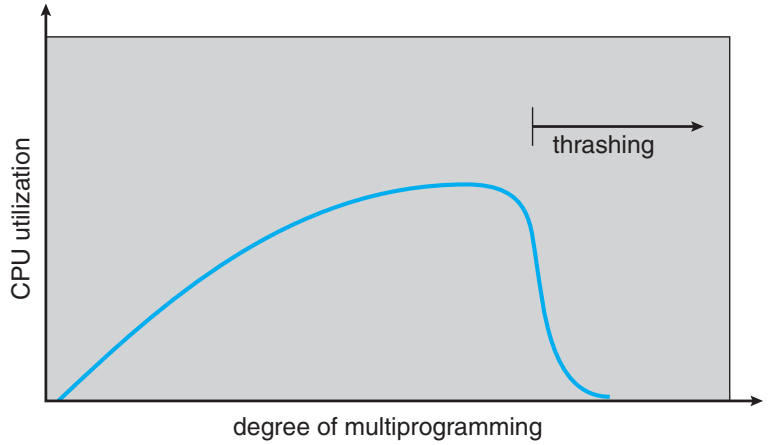
WSClock



Tanenbaum: Figura 4.22

Controle da carga

- *Thrashing*



Silberschatz: Figura 9.18

- O Sistema Operacional só se ocupa das tarefas de paginação e escalonamento
- Todos os processos precisam de mais memória
- Swap (como escolher quais processos vão para o disco?)

Resumo

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

Tanenbaum: Figura 4.23