

Sistemas Operacionais

Gerenciamento de Memória

Memória Virtual

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

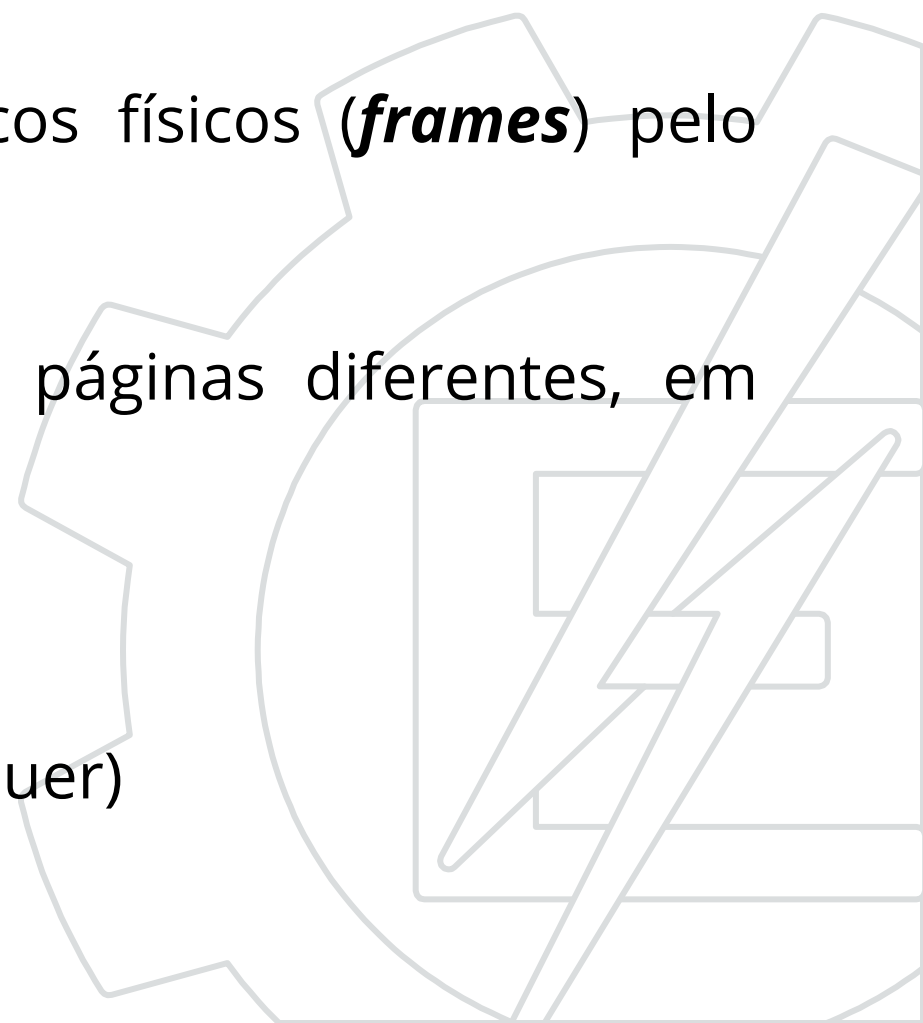


- A segmentação permite que o espaço de endereçamento físico de um processo seja não-contíguo. A paginação é outro esquema de gerenciamento da memória que oferece essa vantagem.
- No entanto, a paginação evita a fragmentação externa e a necessidade de compactação, enquanto a segmentação não faz isso.
- Ela também resolve o considerável problema de acomodar trechos de memória de vários tamanhos na memória de retaguarda

Paginação



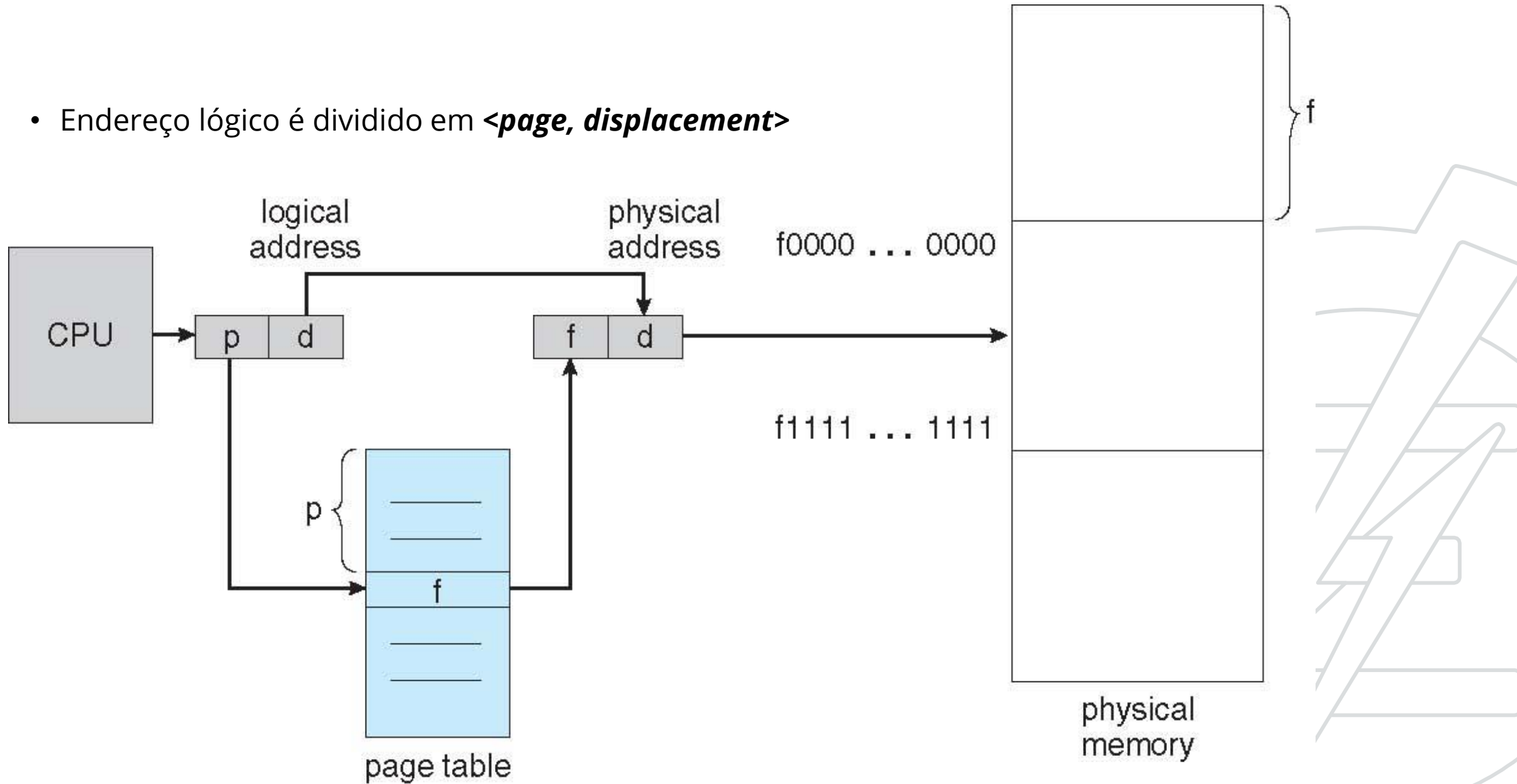
- Chama-se paginação a técnica utilizada para transferência de dados inativos da memória principal para a secundária (*Page-Out*) e transferência de dados ativos de volta para a memória principal (*Page-In*).
- Algumas páginas não podem ser removidas da memória principal a exemplo daquelas que armazenam:
 - a tabela de paginação;
 - rotinas de tratamento de interrupção; e
 - informações manipuladas pelos dispositivos com recurso de acesso direto à memória (DMA).

- Memória (lógica/física) é dividida em blocos de tamanho fixo – potências de 2; p.ex.: 4 KB.
 - Blocos lógicos (***pages***) são mapeadas em blocos físicos (***frames***) pelo *hardware*
 - Endereços lógicos contíguos podem estar em páginas diferentes, em quadros não contíguos
 - Quadros vazios são gerenciados
 - Programa de n páginas requer n quadros (quaisquer)
 - Fragmentação interna (na última página)
- 

Gerenciamento de Memória

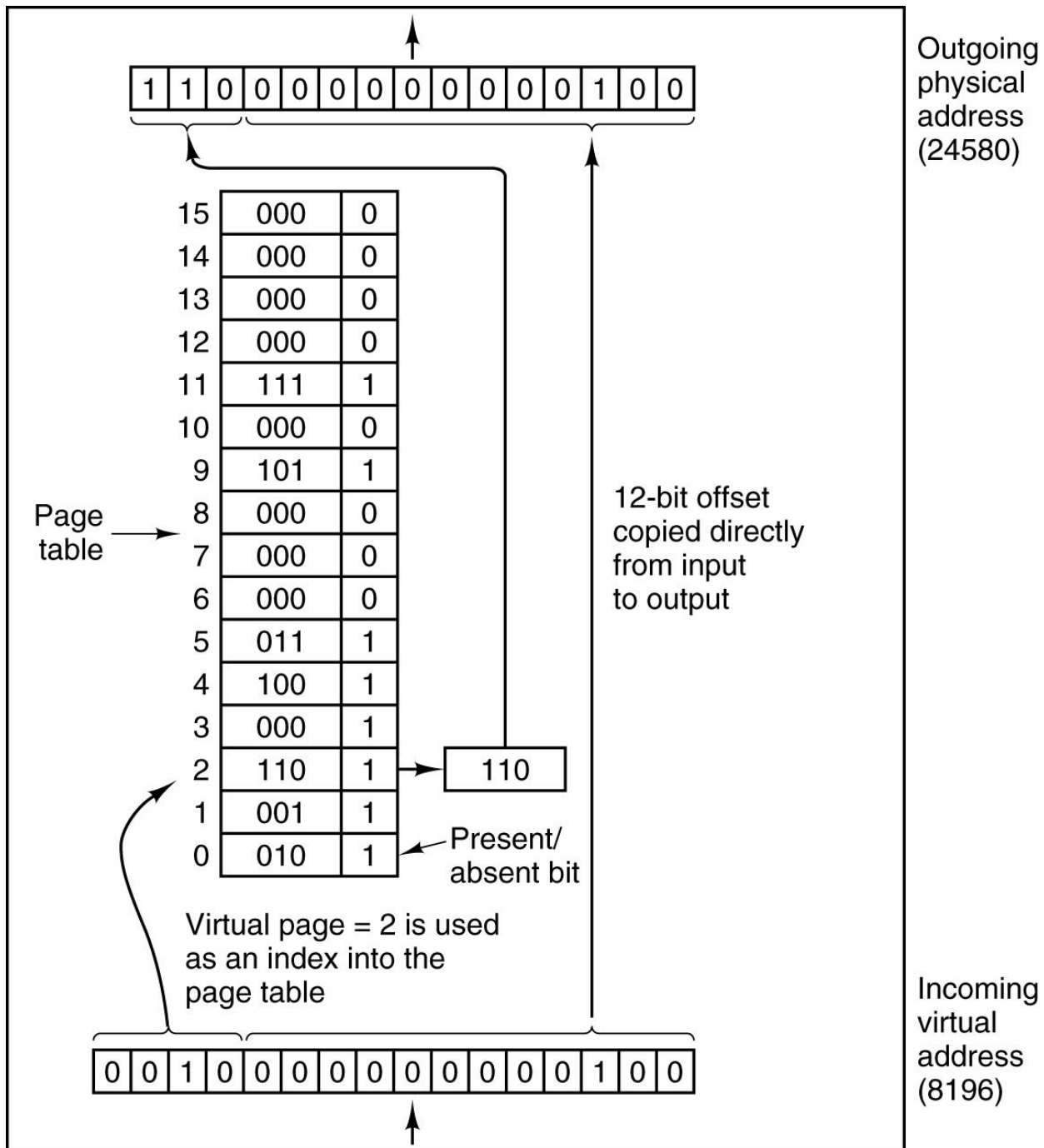
Paginação – Estrutura de tradução de endereços

- Endereço lógico é dividido em **<page, displacement>**



Gerenciamento de Memória

Paginação – Estrutura de tradução de endereços



- MMU com 16 páginas de 4Kb.
- Endereço virtual de 16 bits.
- *Hardware* com 8 frames

- **Páginas:** unidades de tamanho fixo no dispositivo secundário
- **Frames:** unidades correspondentes na memória física (RAM)
- **Page fault:** é o evento quando uma página que não está na RAM é referenciada – Utiliza uma *trap* para carregar ou substituir uma página.
- **Tabela de páginas:** estrutura para mapear uma página ao *frame* correspondente – cada processo possui uma.

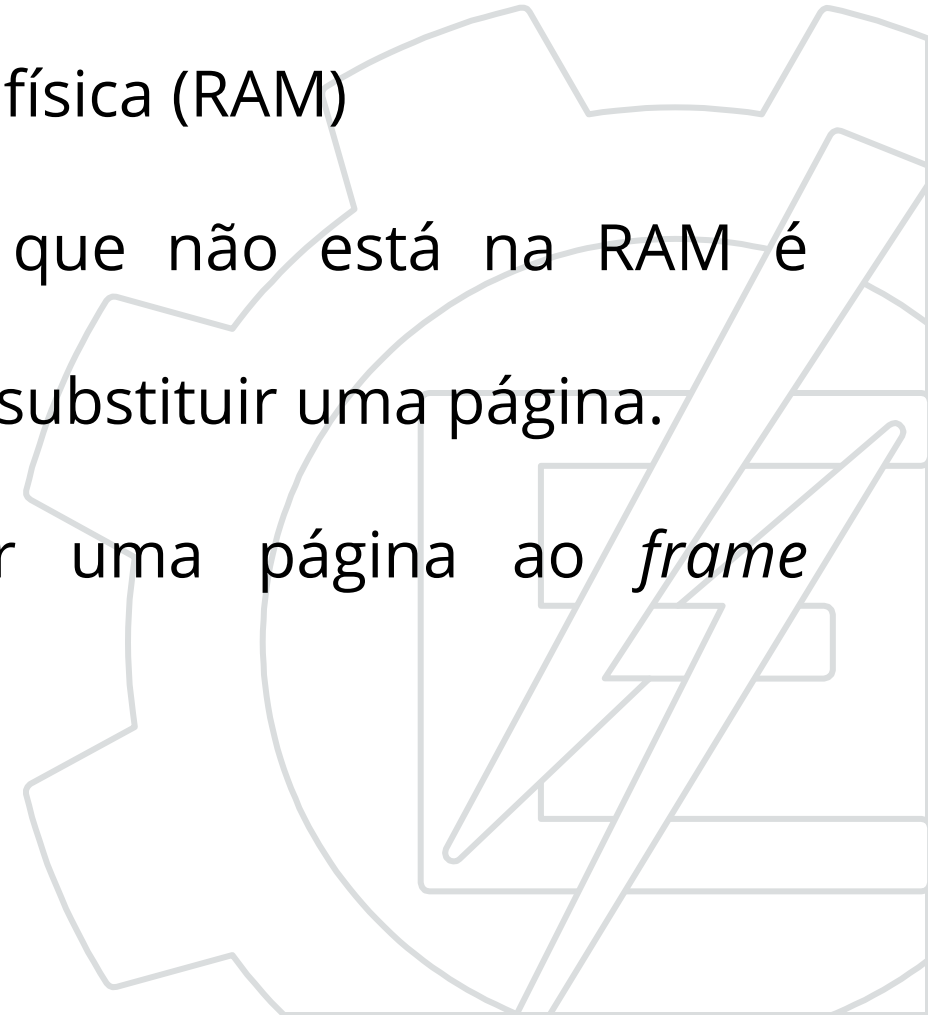


Tabela de páginas



- Onde armazenar as tabelas?
 - **Array de registradores:** se a memória for pequena. Mantidos no *hardware*.
 - **Memória RAM:** a MMU gerencia utilizando alguns registradores.
 - **Cache na MMU:** chamada de **memória associativa** (TLB).
 - TLB - *Translation Lookaside Buffer*
 - Utilizada para melhorar o desempenho da tabela na RAM.

Tabela na RAM

- Utiliza dois registradores:
 - Registrador de base da tabela de página (**PTBR**)
 - Aponta para o início da tabela, indicando o endereço físico da memória onde a tabela está alocada.
 - Registrador de tamanho da tabela de página (**PTLR**)
 - Indica o tamanho (número de entradas) da tabela de páginas → número de páginas.

Tabela na RAM

- Problema:
 - Dois acessos para instrução/dados na RAM
 - Um acesso para a tabela e outro para o dado/instrução em si

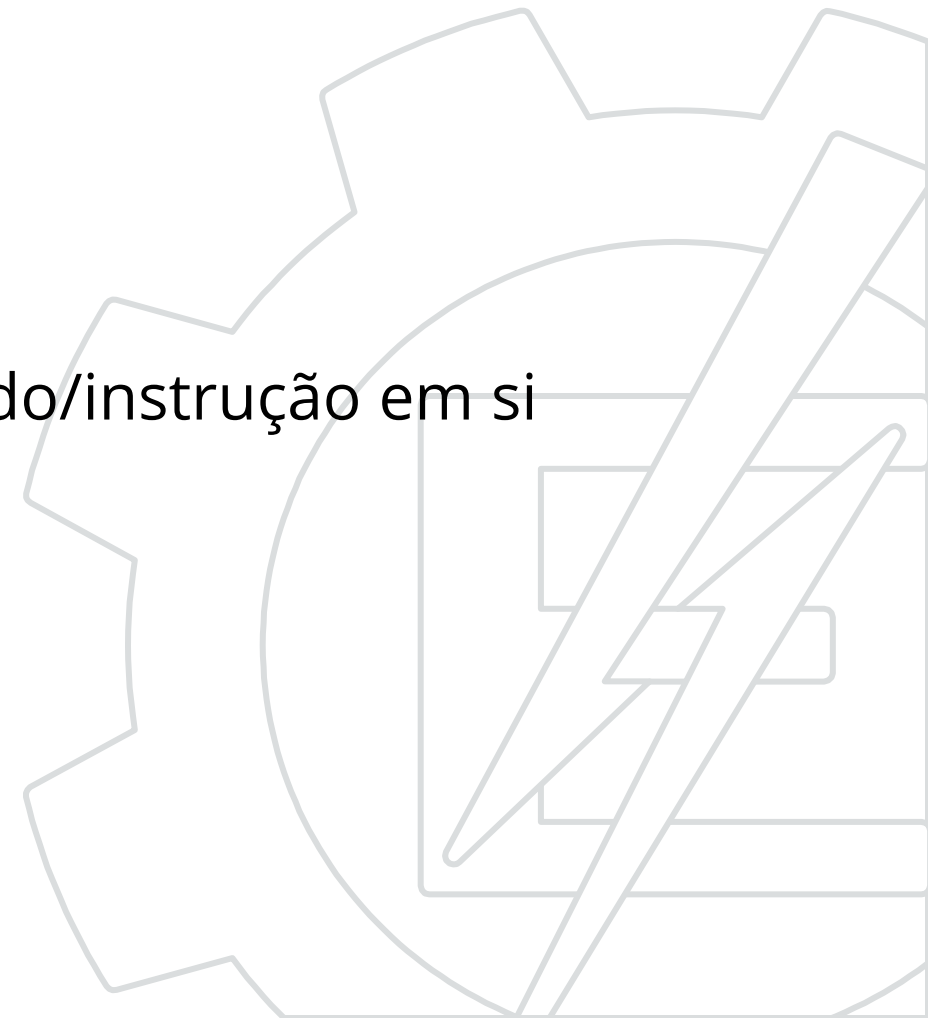
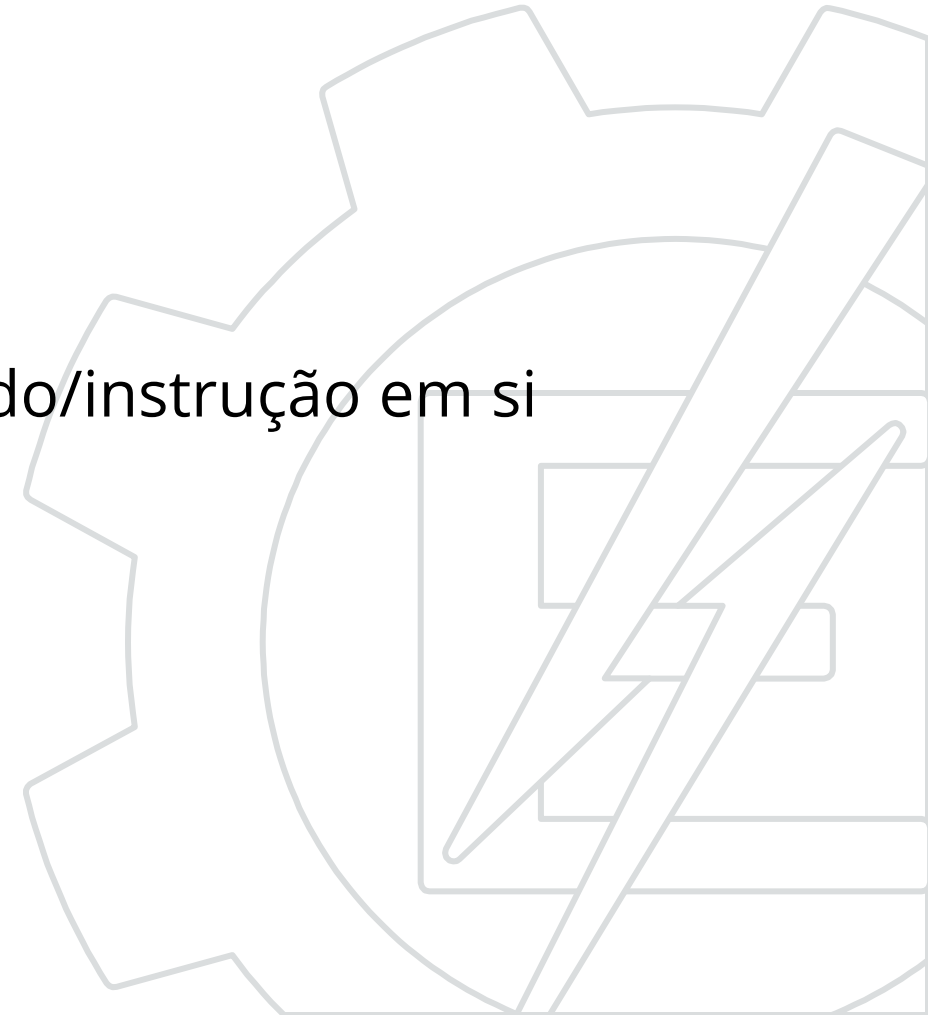


Tabela na RAM

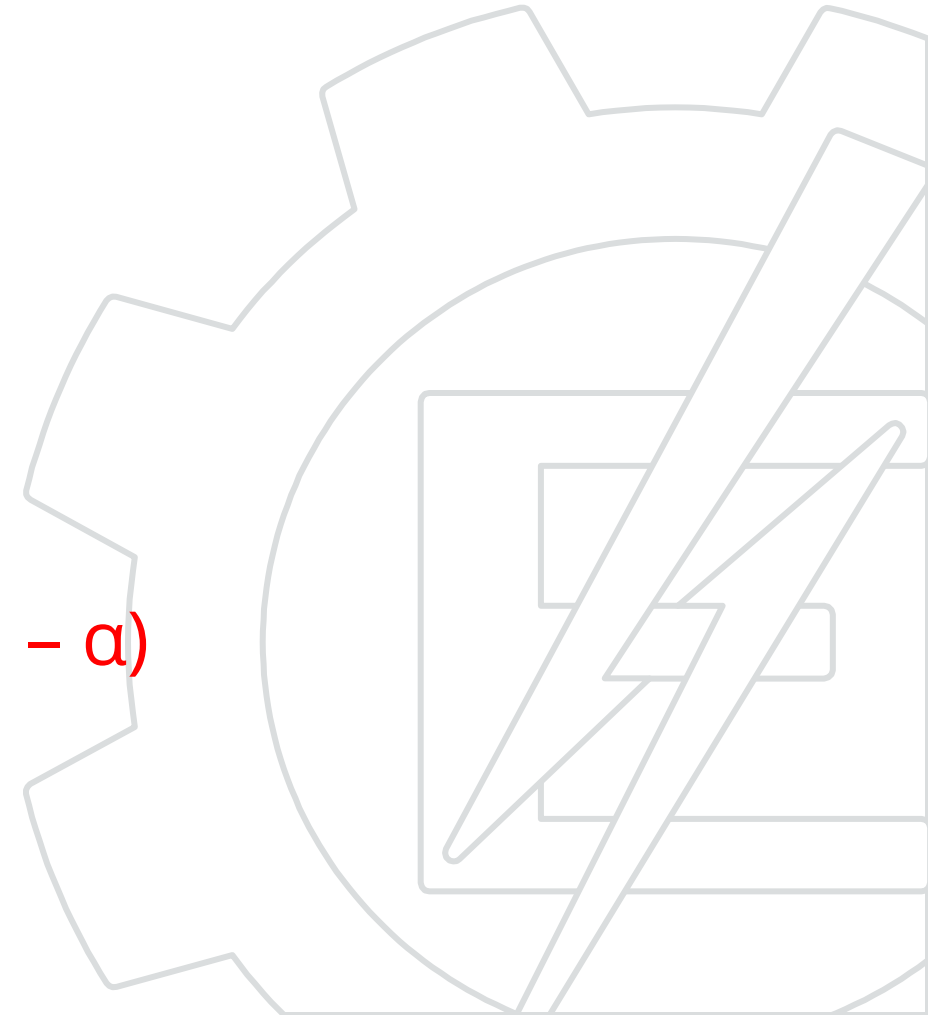
- Problema:
 - Dois acessos para instrução/dados na RAM
 - Um acesso para a tabela e outro para o dado/instrução em si
 - Solução: *cache* denominado **TLB**.



- Consulta à TLB: ϵ
- Tempo de um ciclo de memória: t
- Taxa de acerto (*hit ratio*) na TLB: α
- Tempo de acesso efetivo (TAE):

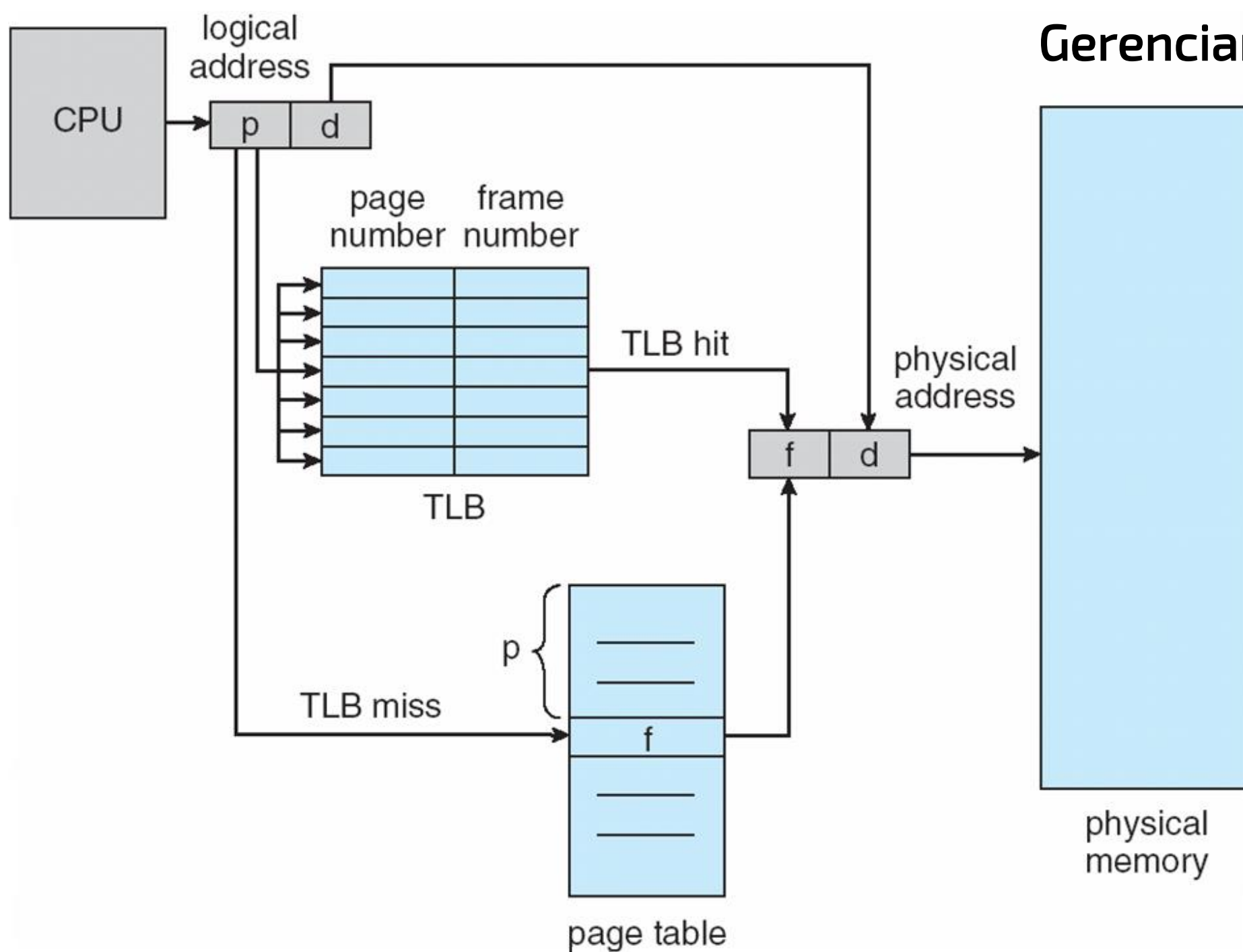
$$\text{TAE} = (t + \epsilon) \alpha + (2t + \epsilon) (1 - \alpha)$$

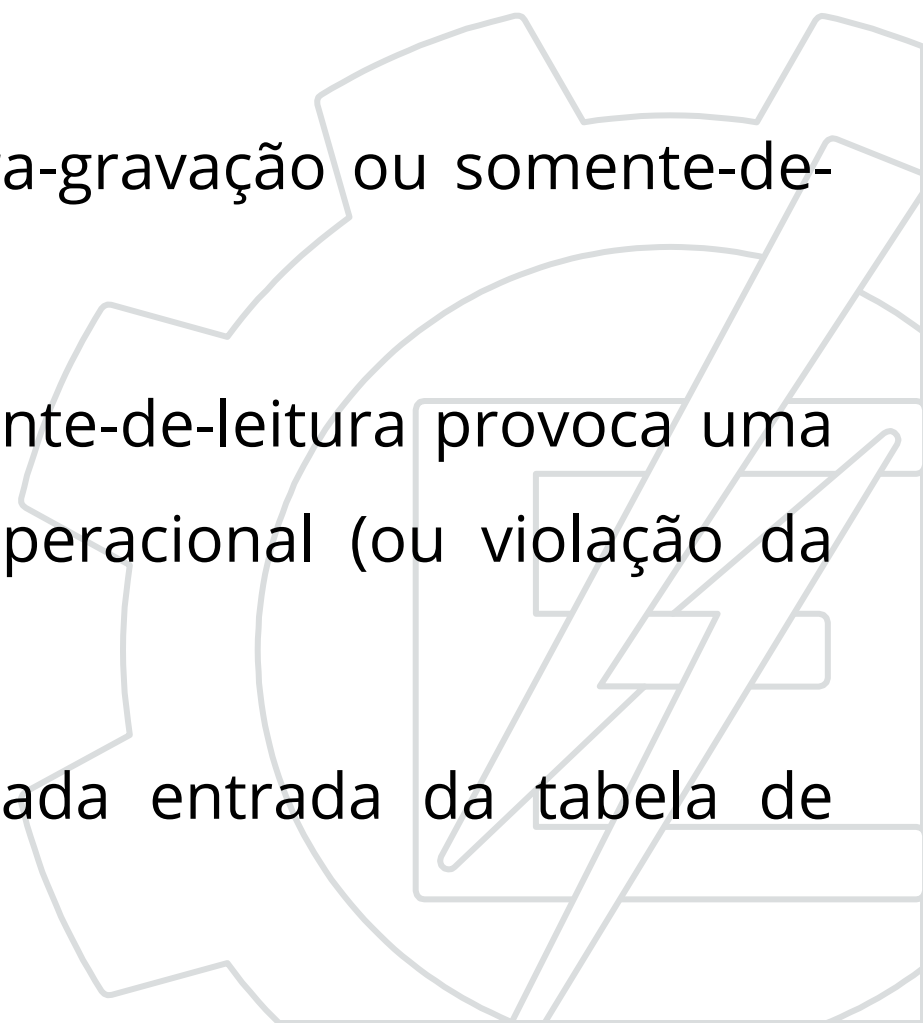
$$\text{TAE} = (2 - \alpha) t + \epsilon$$



Gerenciamento de Memória

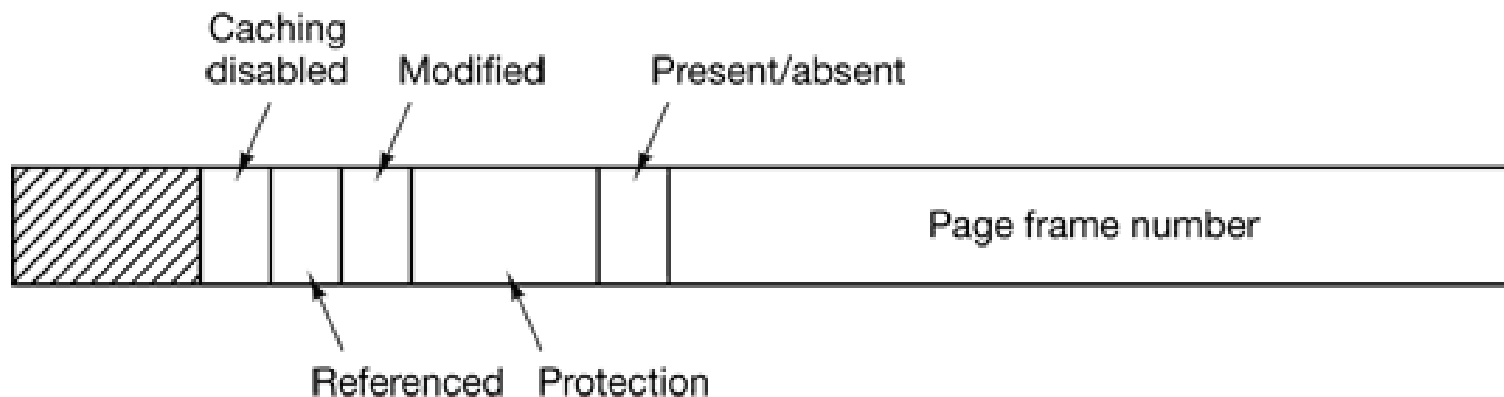
HW de Paginação com TLB



- A proteção da memória em um ambiente paginado é executada por **bits de proteção associados a cada quadro**.
 - Um bit pode definir se uma página é de leitura-gravação ou somente-de-leitura.
 - A tentativa de gravação em uma página somente-de-leitura provoca uma interceptação de hardware para o sistema operacional (ou violação da proteção à memória).
 - Um bit adicional é geralmente anexado a cada entrada da tabela de páginas: um bit válido-inválido.
- 

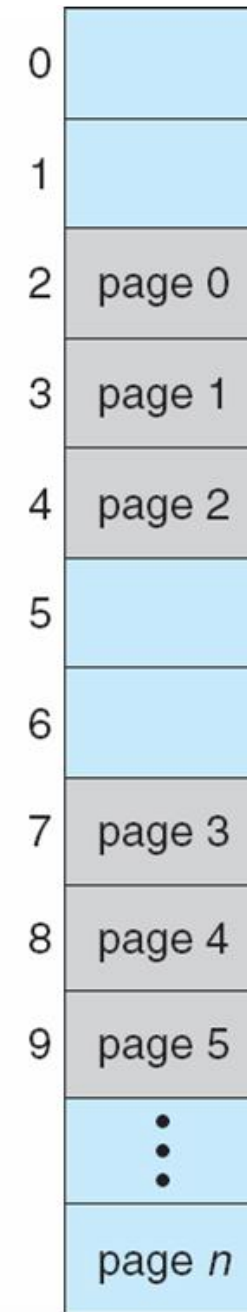
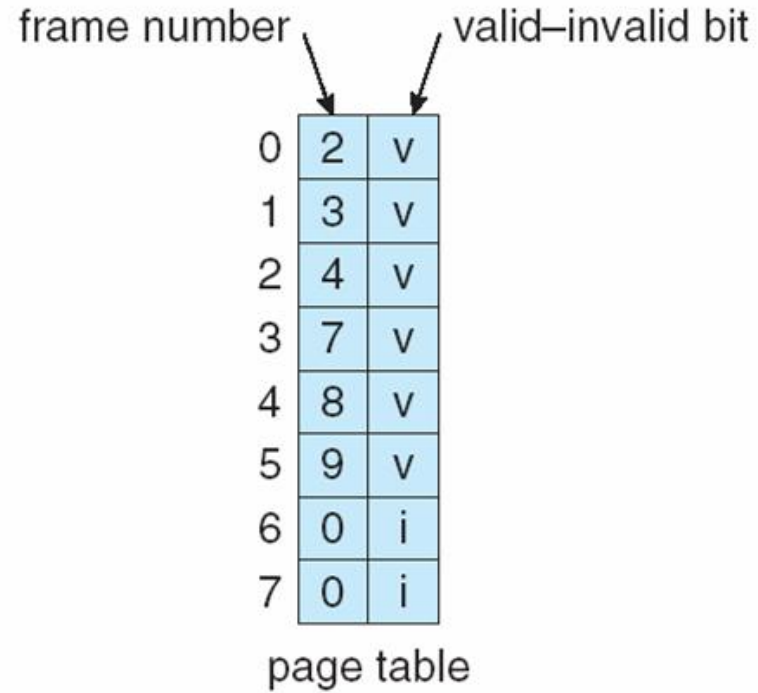
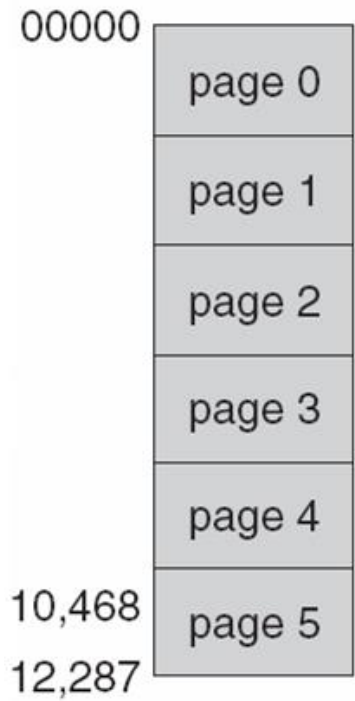
- Componentes

- a) *Page frame number*: identifica o número da página real
- b) Bit de **residência**: se 1 → a página é válida e está presente na RAM.
 - Se 0 → ocorre um *Page fault*.
- a) Bit de **proteção**: 0 → *leitura/escrita*, 1 → *leitura*, 2 → *execução*
- b) Bit de **modificação**: 1 → página alterada, 0 → página não-alterada
- c) Bit de **referência**: 1 → foi referenciada “recentemente”
- d) Bit de **cache**: permite desabilitar o *caching* da página.



Gerenciamento de Memória

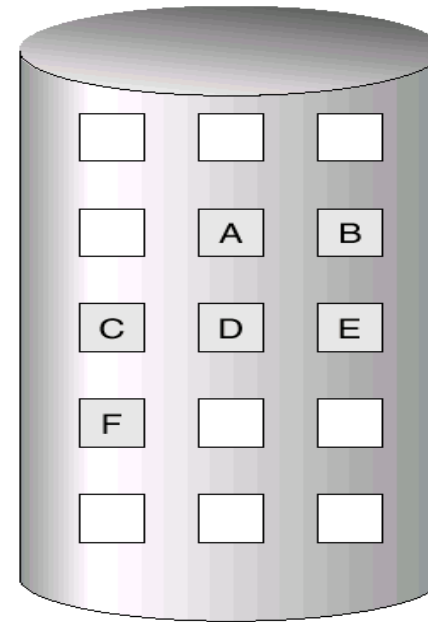
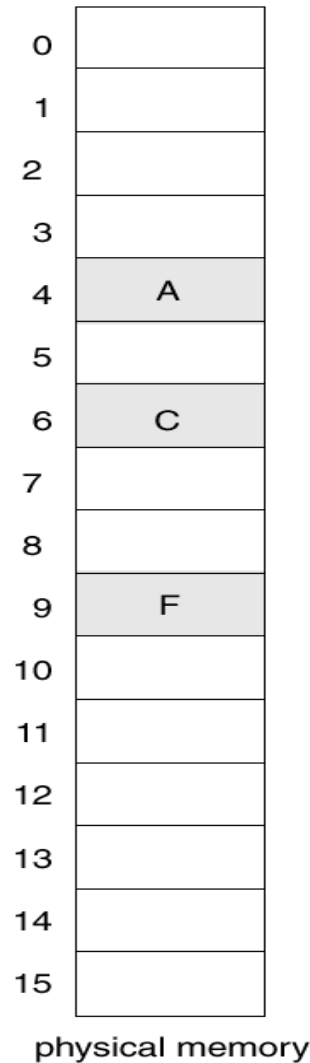
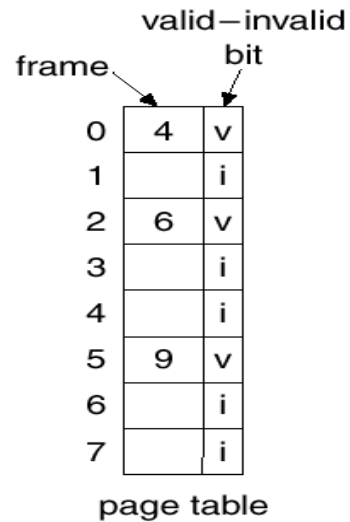
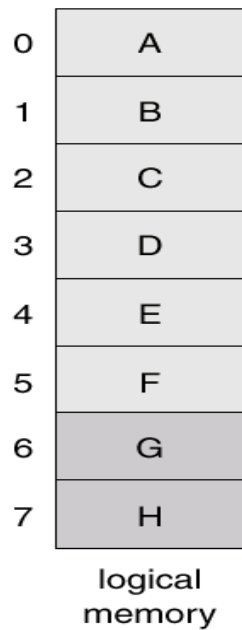
Paginação – Proteção de memória



Gerenciamento de Memória

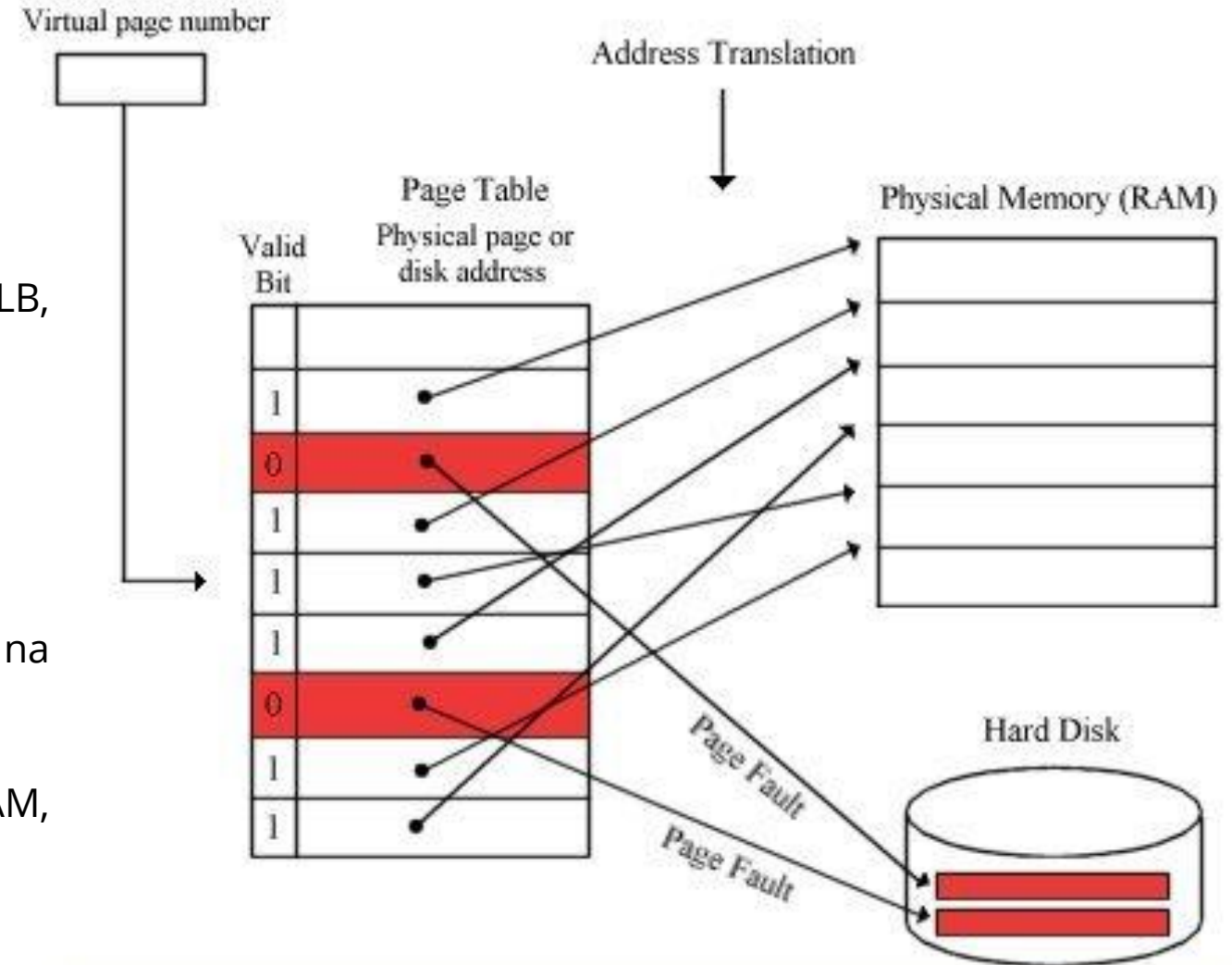
Paginação – Proteção de memória

- O bit válido/inválido é usado para controlar a presença (ou não) da página (lógica) na memória física (principal)



Tipos de *page faults*

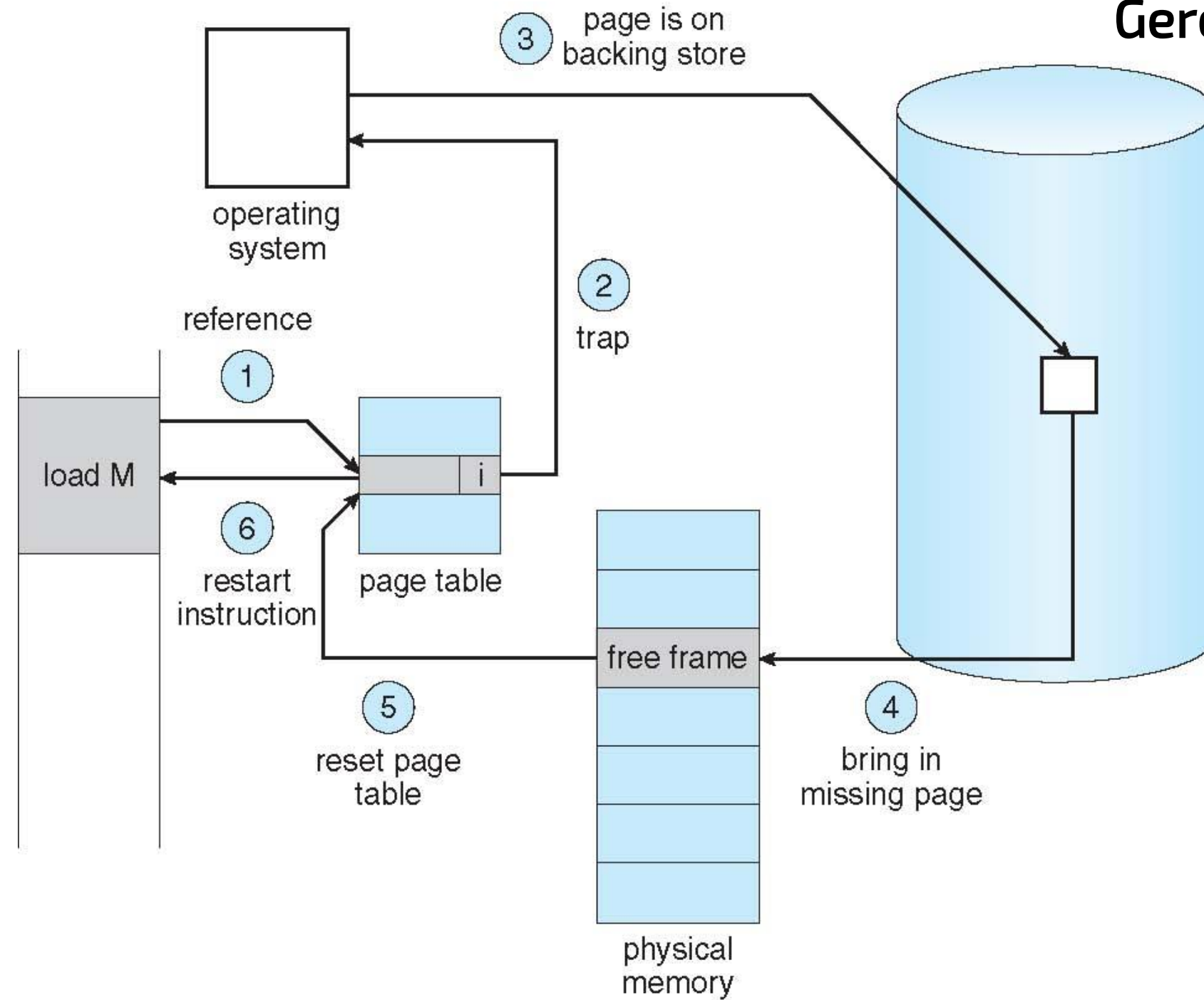
- *Soft miss*:
 - Quando a página referenciada não está na TLB, mas está na RAM
 - Basta atualizar a TLB.
- *Hard miss*:
 - A página não está na memória física, nem na TLB.
 - É necessário trazer a página do disco à RAM, então à TLB.



The page table maps each page in virtual memory to either a page in RAM or Hard Disk.

Gerenciamento de Memória

Falha de páginas



Como organizar tabelas de páginas?

- Problema com tabelas de páginas grandes
- Resultado de RAMs atuais de grande capacidade
- Estruturas:
 - Tabela hierárquica (multinível)
 - Tabela *hash*
 - Tabela invertida



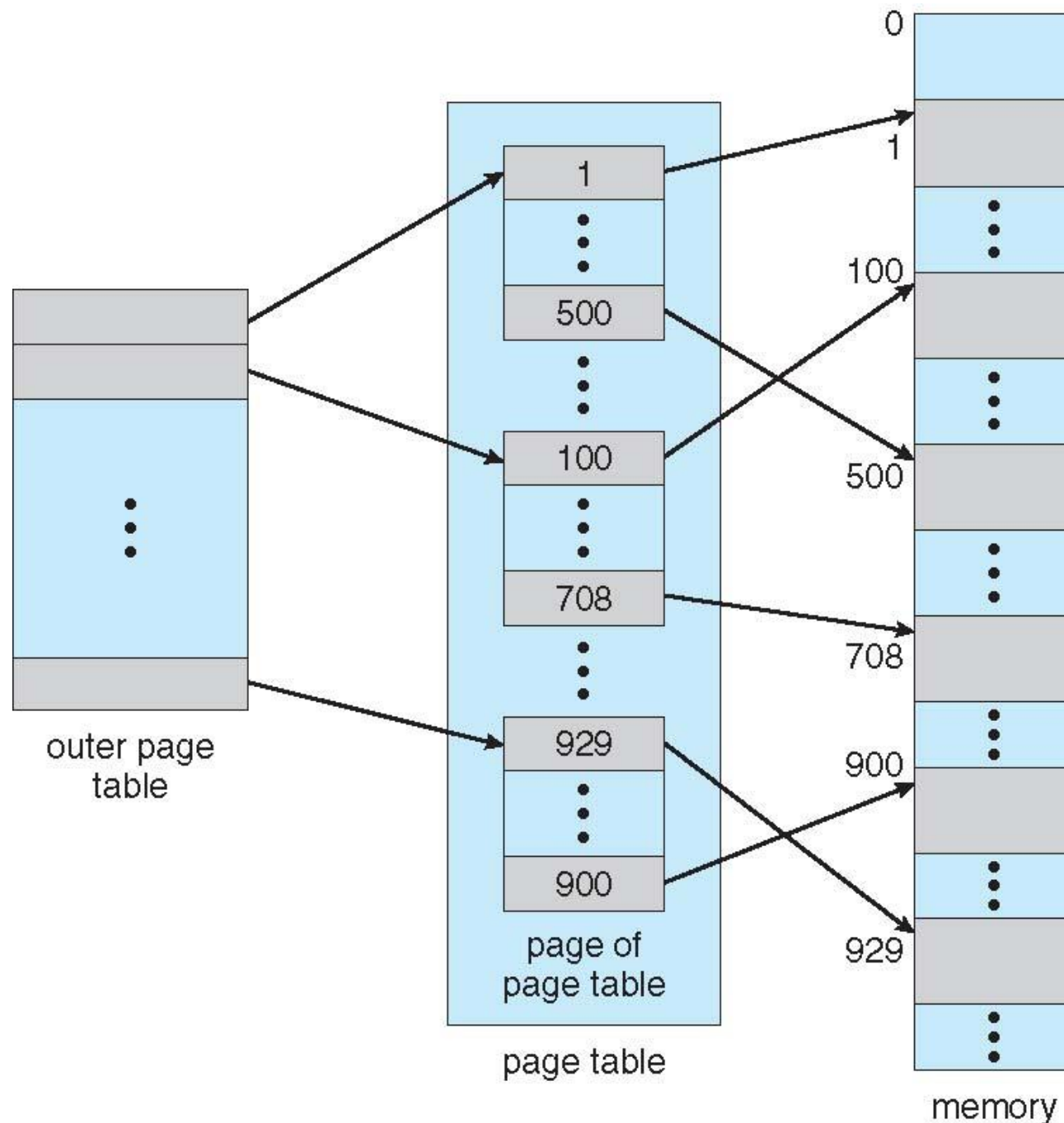
Tabela de páginas

Hierárquica ou multinível



Gerenciamento de Memória

Tabela de páginas hierárquica



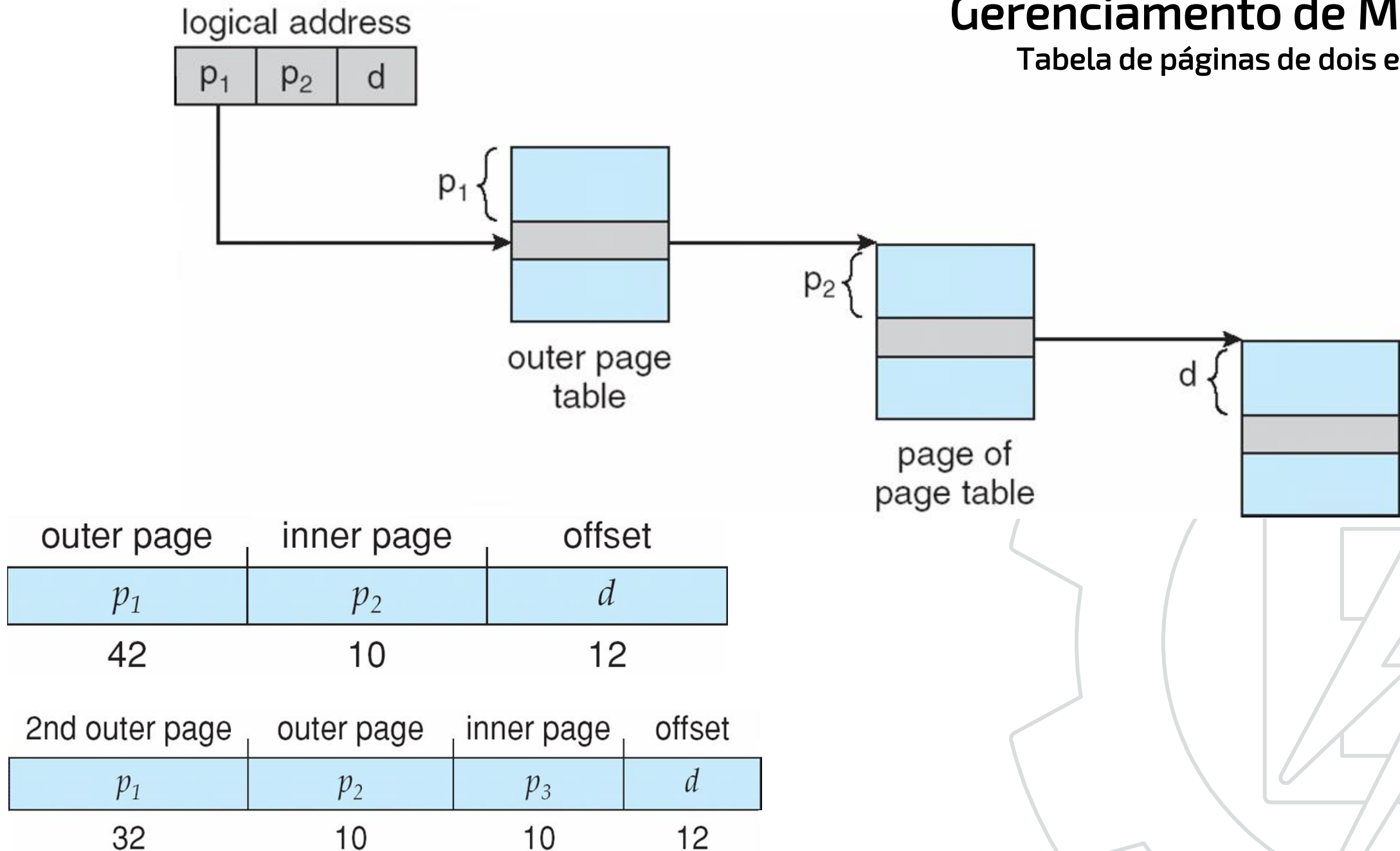
- Quebra do espaço de endereços lógicos em múltiplas tabelas de páginas.
- Mantém apenas a parte necessária da tabela.
- Ocupa menos espaço para o próprio Sistema Operacional.
- Uma técnica simples é uma **tabela de páginas em dois níveis**.

- Páginas não são muito grandes (4 KB → 12 bits)
- Se o resto do endereço identificar a página
 - Com endereços de 32 bits pode haver mais de 1 milhão de páginas (20 Bits)
 - Tabela gigantesca teria que ser alocada em posições contíguas da memória!
- Solução: quebrar o endereço em várias tabelas
 - ex.: para 32 bits, criar outros dois níveis de tabelas - cada um com 10 bits

$$2^{32} = 4.294.967.296$$

Gerenciamento de Memória

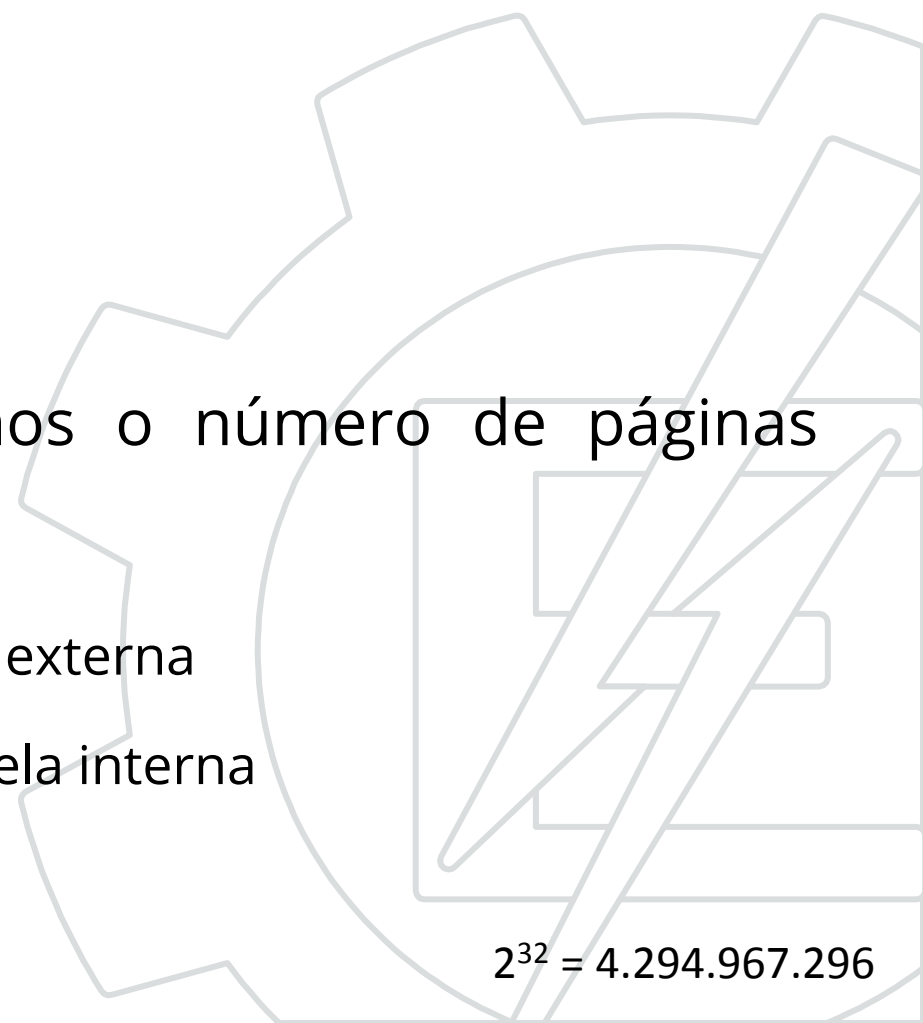
Tabela de páginas de dois e três níveis



Gerenciamento de Memória

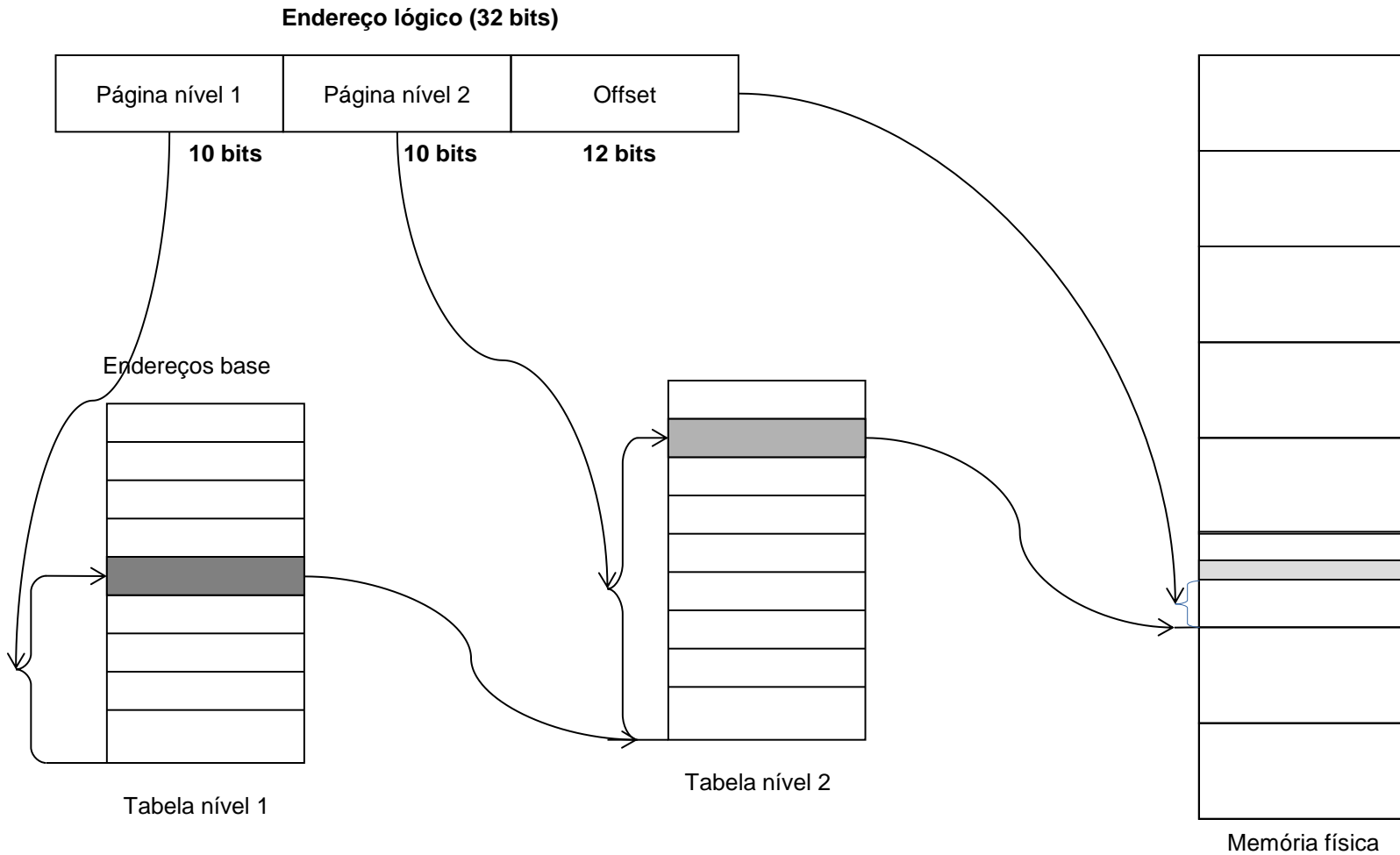
Tabela de endereçamento com dois níveis

- Um endereço lógico em máquinas de 32 bits é dividido em:
 - Um número de página contendo 20 bits
 - Um deslocamento de página contendo 12 bits
- Com a paginação da tabela de páginas, temos o número de páginas dividido em:
 - Número de páginas PT1 (10 bits): índice da tabela mais externa
 - Número de páginas PT2 (10 bits): deslocamento da tabela interna


$$2^{32} = 4.294.967.296$$

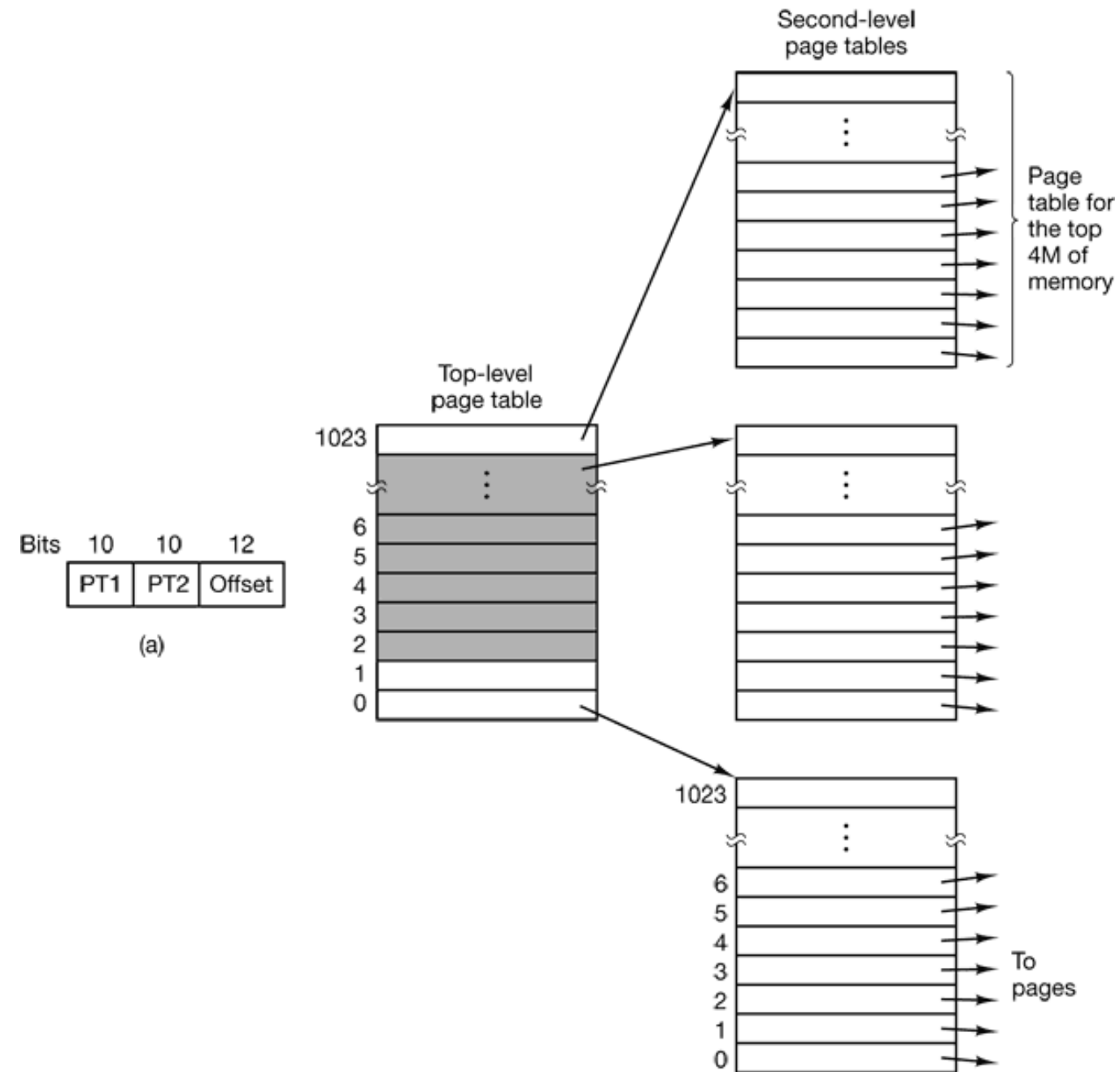
Gerenciamento de Memória

Processo de tradução com dois níveis



Gerenciamento de Memória

Tabela de páginas hierárquica



Gerenciamento de Memória

Tabela de páginas hierárquica

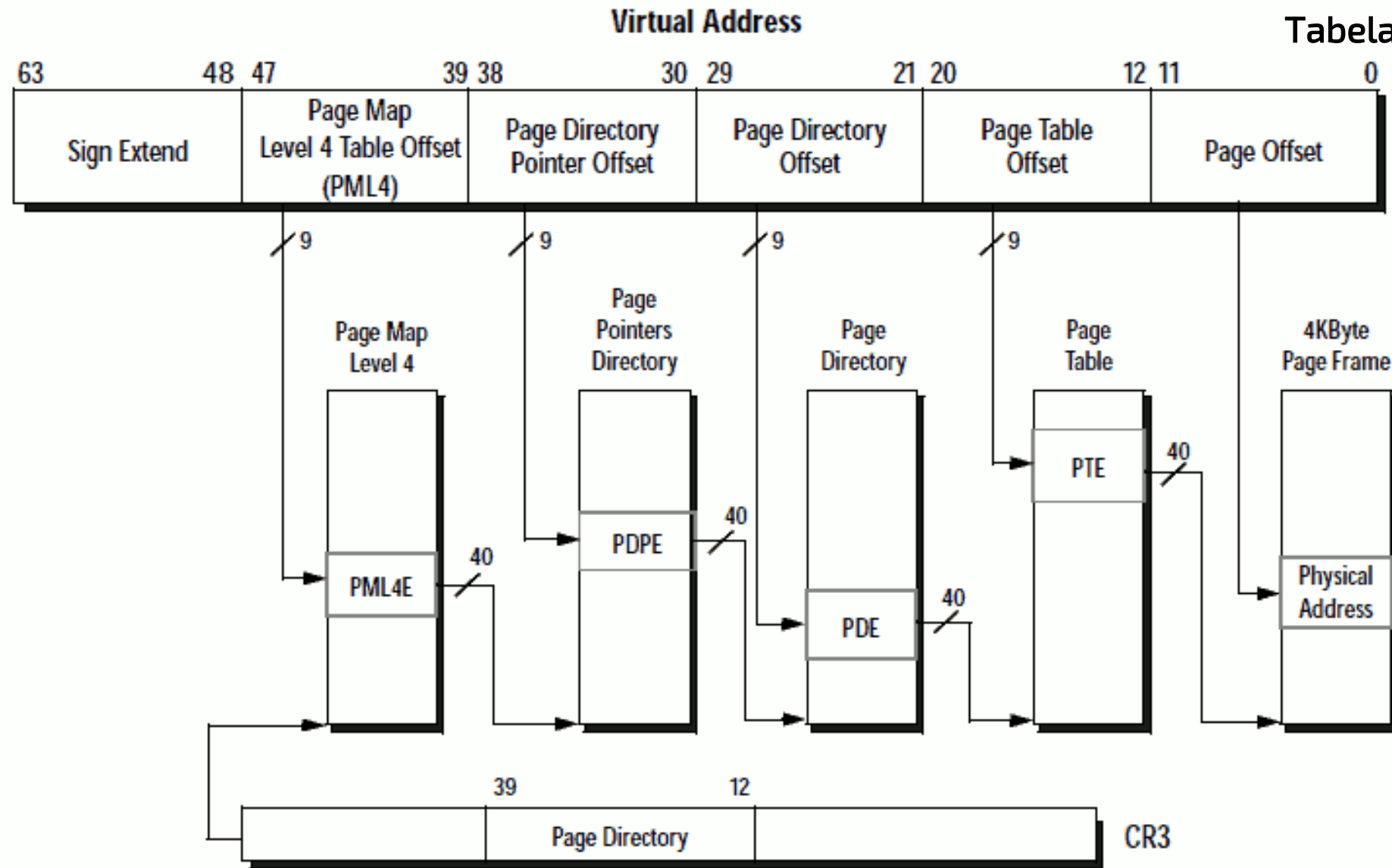


Figure 17. 4KB-Page Translation

$$2^{40} = 1.099.511.627.776$$

Tabela de páginas

por *hash*



Gerenciamento de Memória

Tabela de páginas por *hash*

- Uma abordagem comum para a manipulação de **espaços de endereçamento de 32 bits** é usar uma tabela de páginas com *hash*, com o valor do *hash* sendo o número da página virtual.
- Não funciona tão bem com 64 bits – tabelas de páginas \approx 60 PB
- Cada entrada da tabela com *hash* contém uma lista encadeada de elementos que são mapeados para a mesma localização por uma função *hash* (para manipular as colisões).

Symbol	Prefix	SI Meaning	Binary meaning
k	kilo	$10^3 = 1000^1$	$2^{10} = 1024^1$
M	mega	$10^6 = 1000^2$	$2^{20} = 1024^2$
G	giga	$10^9 = 1000^3$	$2^{30} = 1024^3$
T	tera	$10^{12} = 1000^4$	$2^{40} = 1024^4$
P	peta	$10^{15} = 1000^5$	$2^{50} = 1024^5$
E	exa	$10^{18} = 1000^6$	$2^{60} = 1024^6$
Z	zetta	$10^{21} = 1000^7$	$2^{70} = 1024^7$
Y	yotta	$10^{24} = 1000^8$	$2^{80} = 1024^8$

Gerenciamento de Memória

Tabela de páginas por *hash*

- Cada elemento é composto por três campos:
 1. O número da página virtual;
 2. O valor do quadro (*frame*) de página mapeado; e
 3. Um ponteiro para o próximo elemento na lista encadeada.

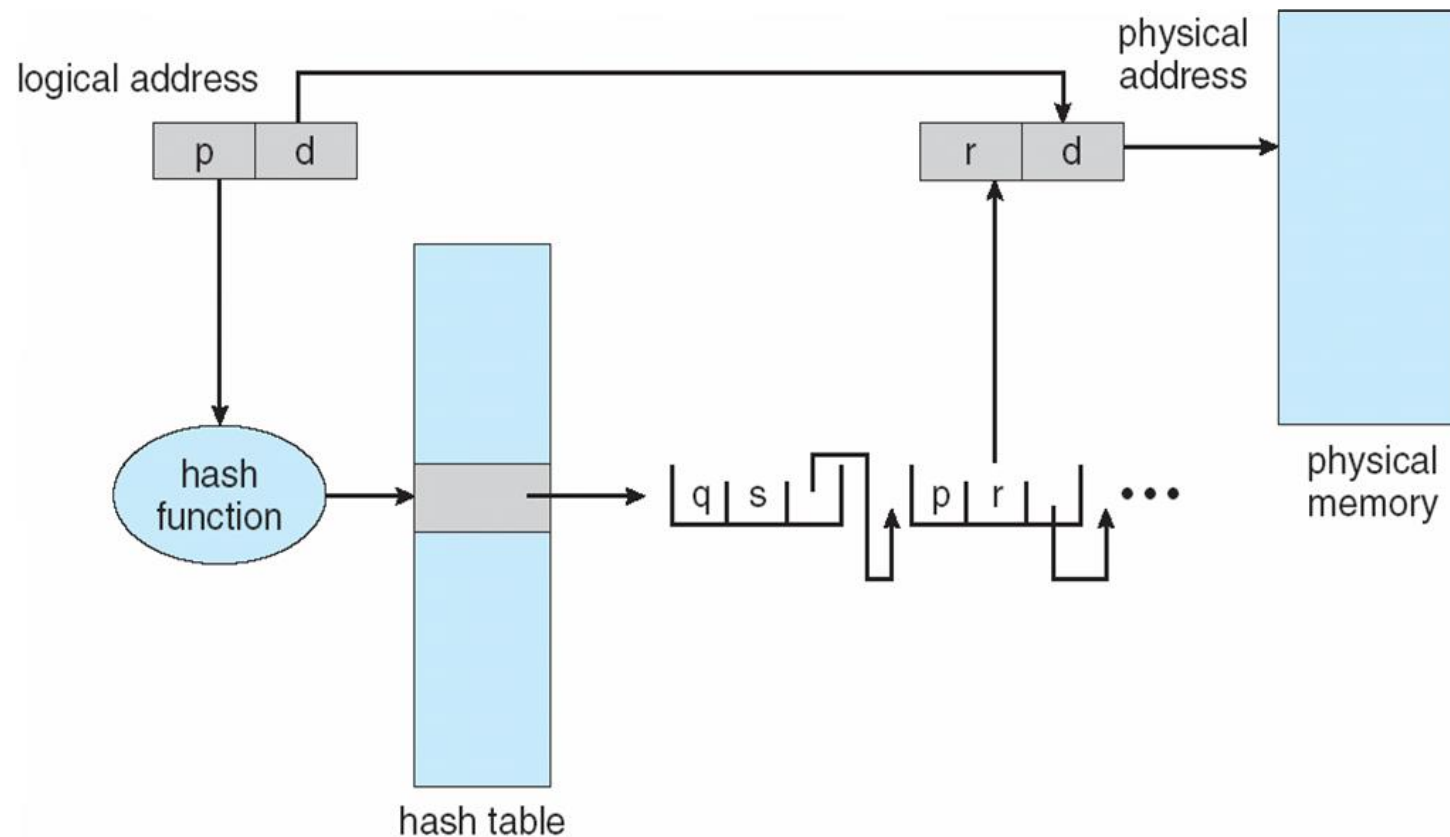


Tabela de páginas

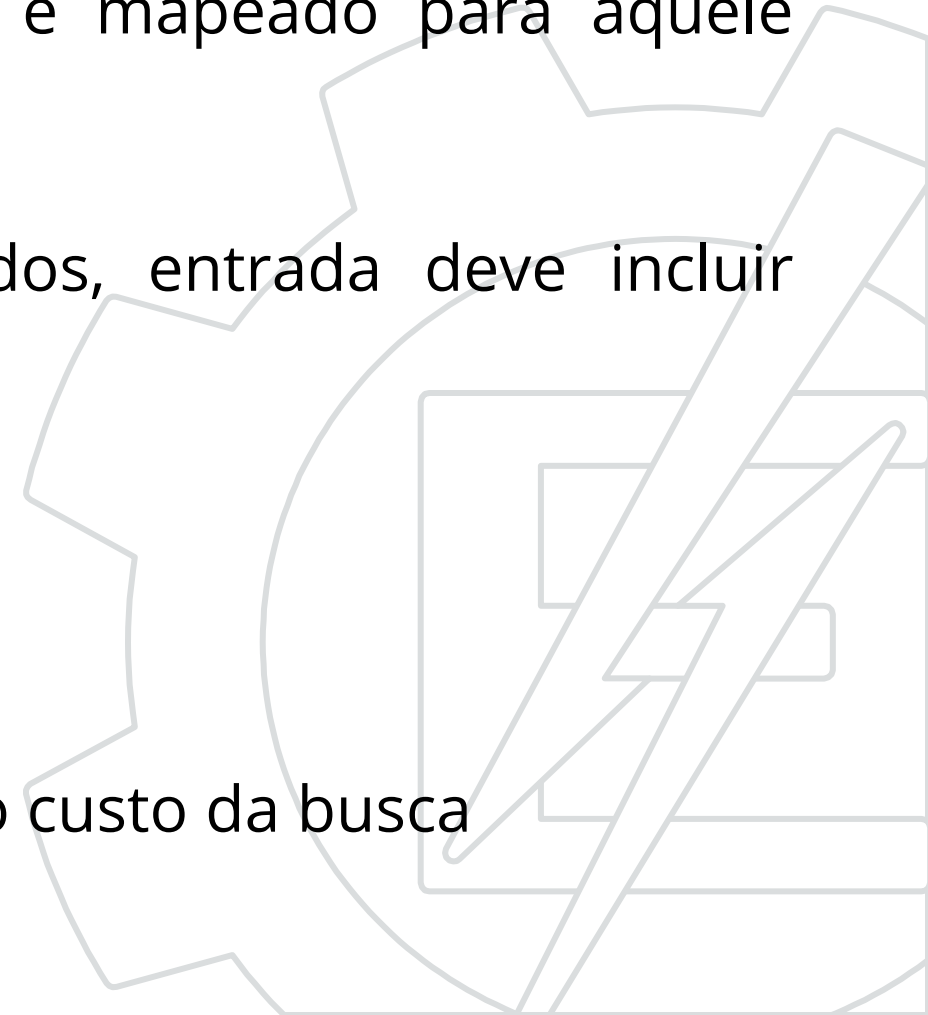
invertida



Gerenciamento de Memória

Tabela de páginas invertida

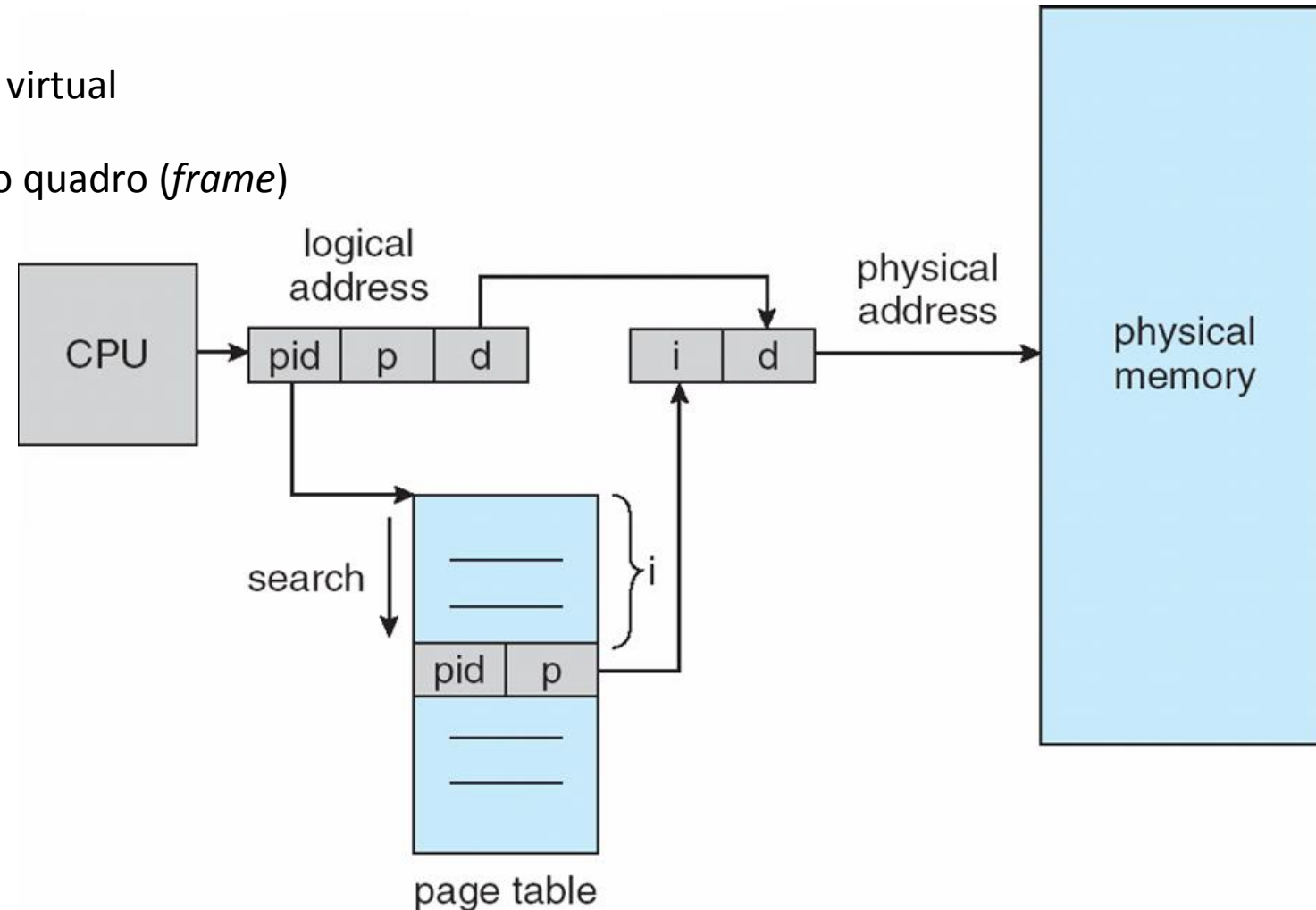
- Uma entrada para cada quadro da memória física
- Entrada na tabela é o endereço lógico que é mapeado para aquele quadro (se não-vazio)
- Como vários processos podem ser carregados, entrada deve incluir identificador do processo
- Reduz a demanda por memória para a tabela
- Aumenta o tempo de acesso
- Pode-se usar uma tabela de *hash* para reduzir o custo da busca



Gerenciamento de Memória

Tabela de páginas invertida

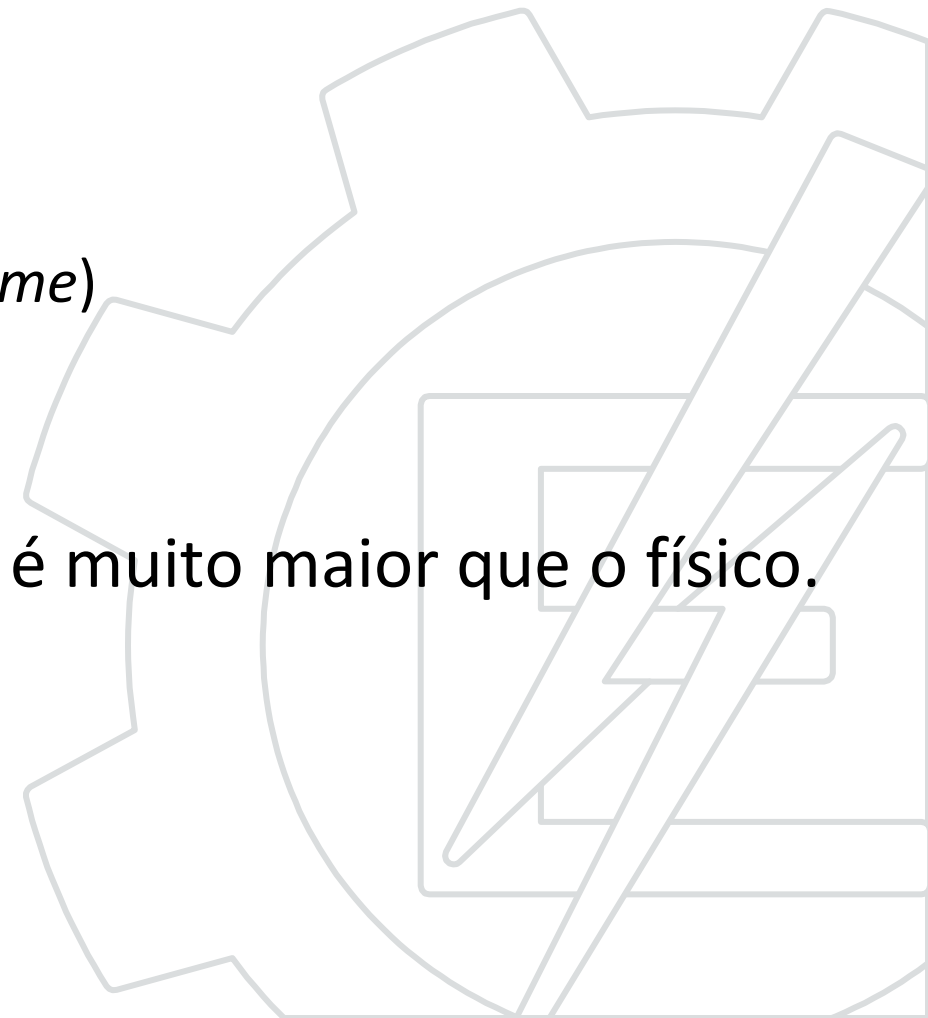
- Possui uma entrada por moldura (*frame*) na memória física, em vez de uma entrada por página no espaço virtual.
- A entrada possui o processo e a página virtual
- O deslocamento na leitura é o índice do quadro (*frame*)



Gerenciamento de Memória

Tabela de páginas invertida

- Possui uma entrada por moldura (*frame*) na memória física, em vez de uma entrada por página no espaço virtual.
- A entrada possui o processo e a página virtual
- O deslocamento na leitura é o índice do quadro (*frame*)
- Poupam muito espaço quando o espaço virtual é muito maior que o físico.



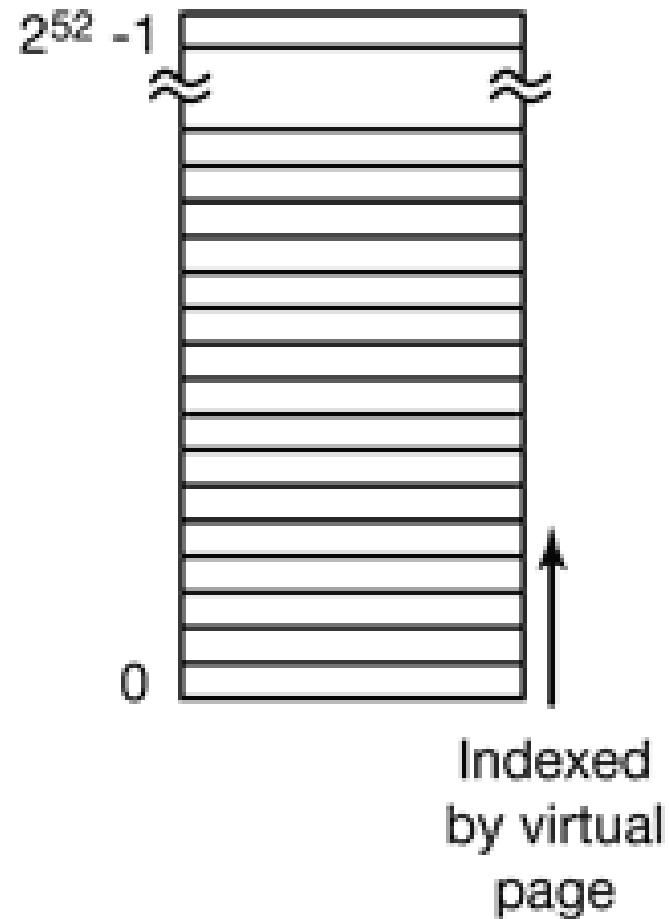
Problemas

- Necessidade de leitura de toda a tabela a cada acesso na memória → Aumenta consideravelmente o tempo de leitura da memória.
- O que fazer?
 - Utilizar **TLB** (memória associativa) para guardar as mais acessadas
 - Caso a página buscada não esteja na TLB, devemos procurar em toda a tabela invertida.

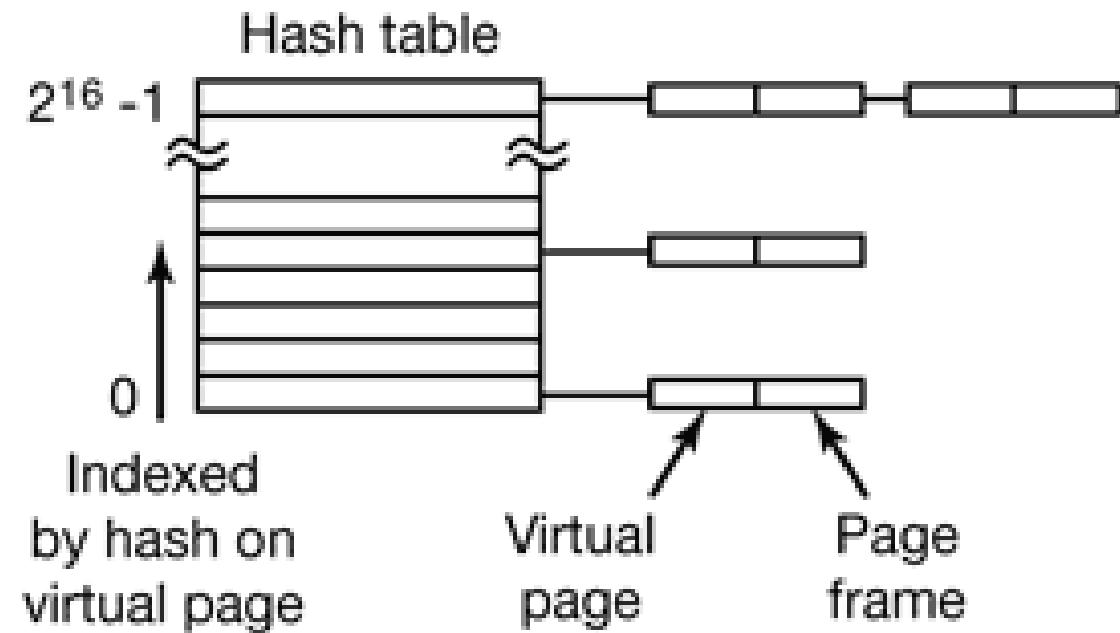
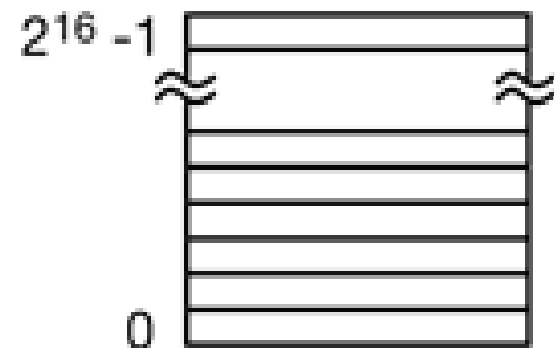
Gerenciamento de Memória

Estrutura da tabela de páginas

Traditional page table with an entry for each of the 2^{52} pages



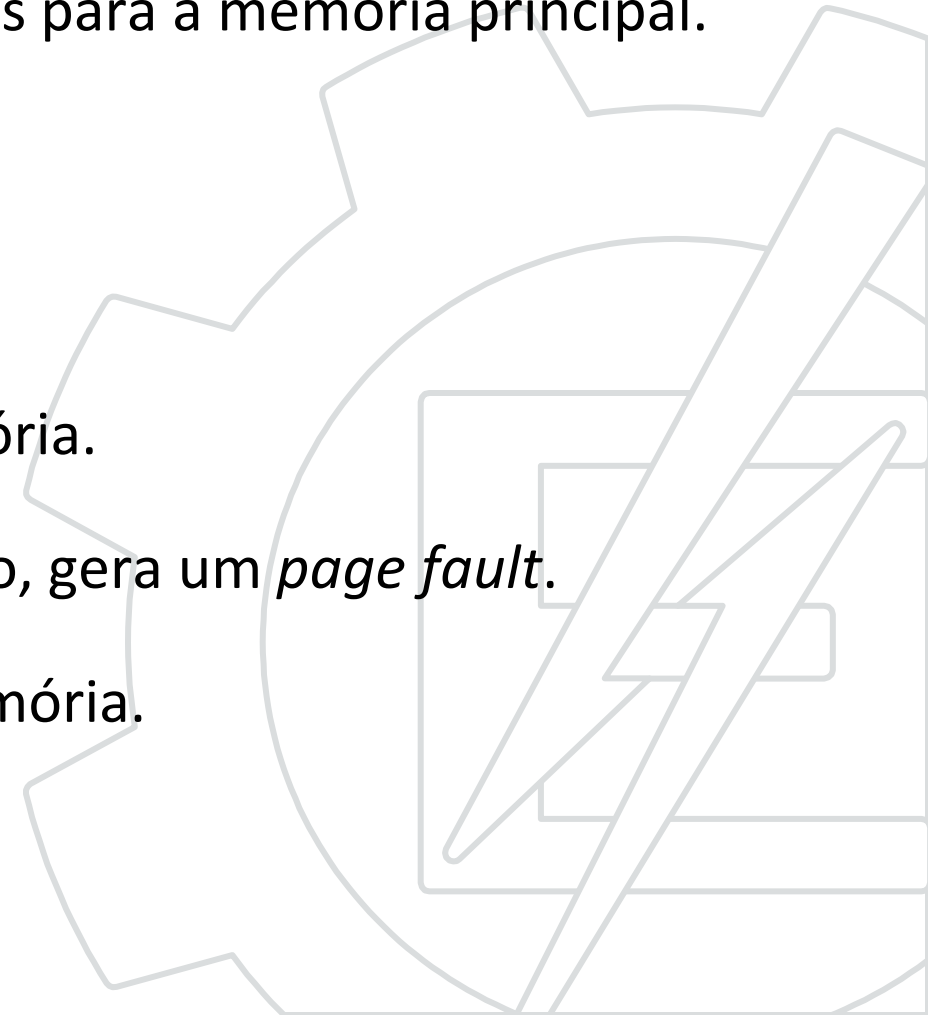
256-MB physical memory has 2^{16} 4-KB page frames



Busca e alocação de páginas



- **Paginação simples:**
 - Todas as páginas virtuais do processo são carregadas para a memória principal.
- **Paginação por demanda (*demand paging*):**
 - Processos começam com nenhuma página na memória.
 - Assim que a CPU tenta executar a primeira instrução, gera um *page fault*.
 - O Sistema Operacional traz a página que falta à memória.



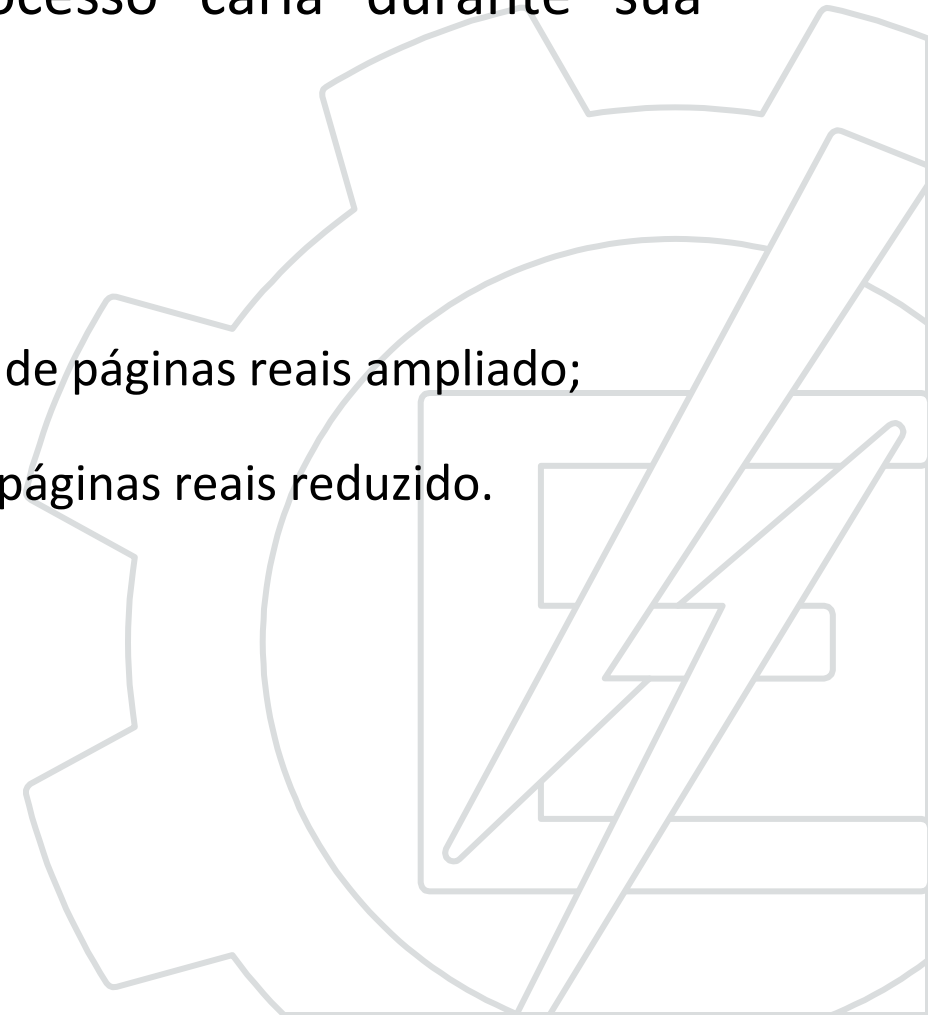
Alocação Fixa

- Cada processo tem um número máximo de páginas reais, definido quando o processo é criado.
- O limite pode ser igual para todos os processos.
- Vantagem: simplicidade
- Desvantagem:
 - Número muito pequeno de páginas reais pode causar muita paginação;
 - Número muito grande de páginas reais causa desperdício de memória principal.



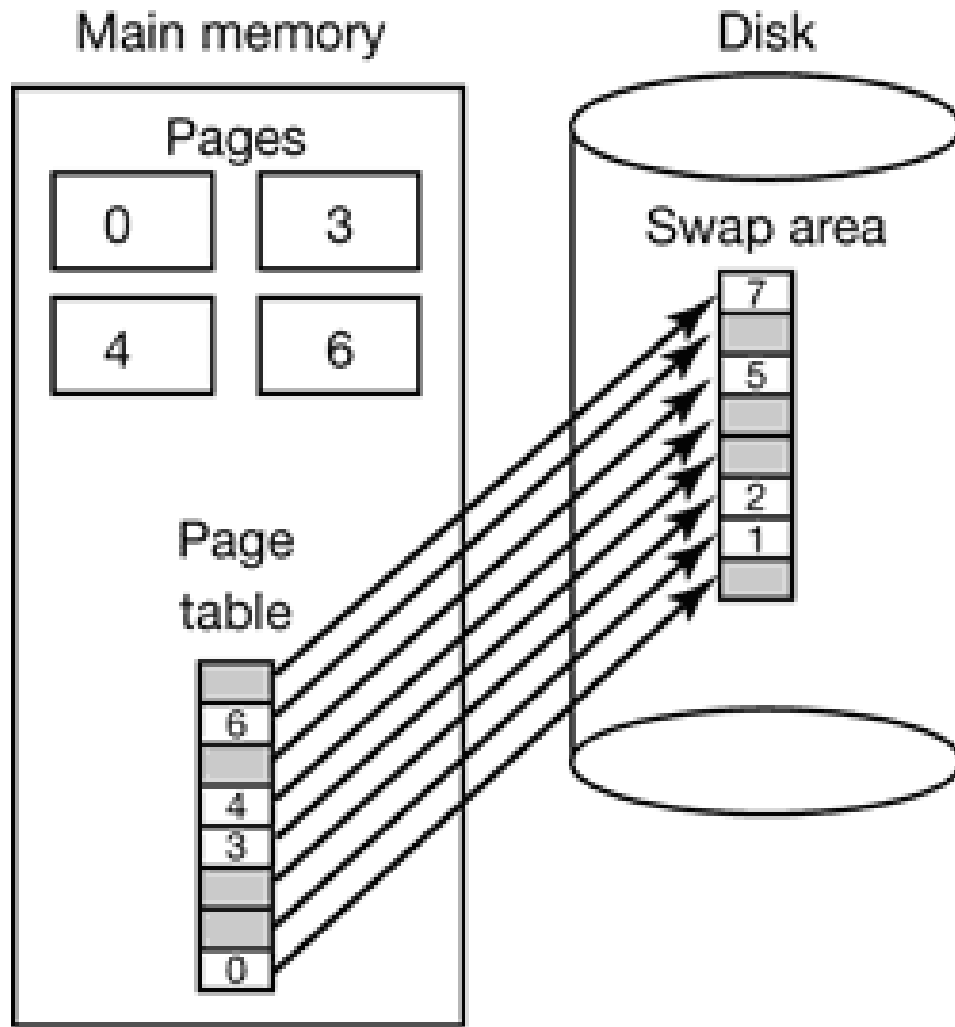
Alocação Dinâmica

- Número máximo de páginas reais alocadas ao processo varia durante sua execução.
- Vantagem:
 - Processos com elevada taxa de paginação podem ter seu limite de páginas reais ampliado;
 - Processo com baixa taxa de paginação podem ter seu limite de páginas reais reduzido.
- Desvantagem:
 - Necessidade de monitoramento constante



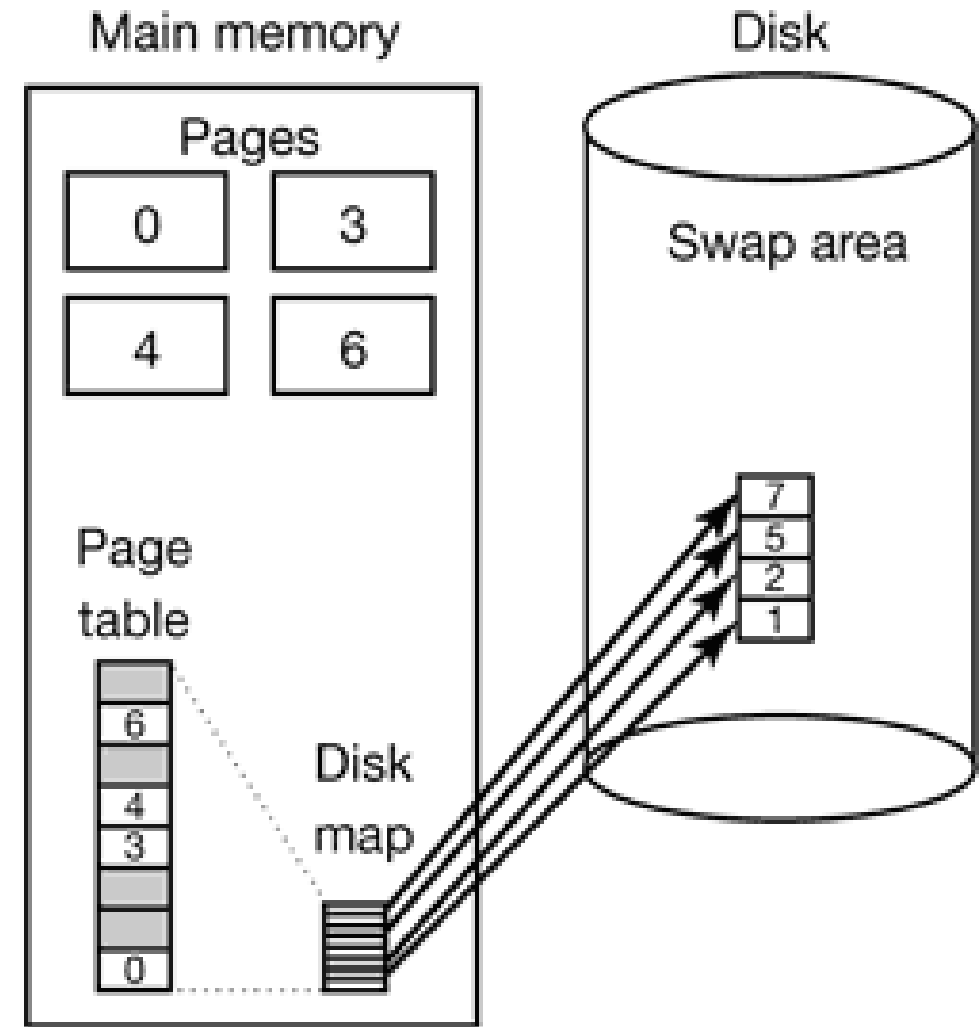
Gerenciamento de Memória

Alocação de páginas



(a)

(a) Alocação fixa ou estática



(b)

(b) Alocação dinâmica

Paginação

Políticas de substituição e
compartilhamento de páginas



Gerenciamento de Memória

Política de substituição de páginas

Local:

- Considera apenas o processo em questão.



	Idade
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

Quanto maior o índice Idade, mais recentemente a página foi utilizada.

Gerenciamento de Memória

Política de substituição de páginas

Local:

- Considera apenas o processo em questão.

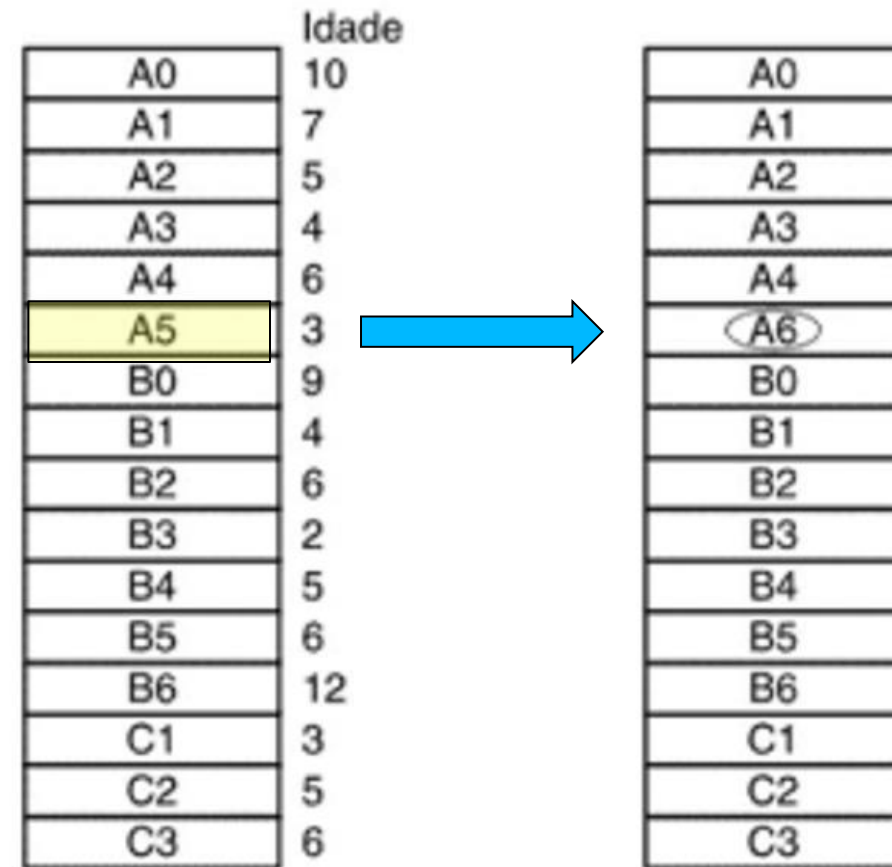
	Idade
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

Gerenciamento de Memória

Política de substituição de páginas

Local:

- Considera apenas o processo em questão.

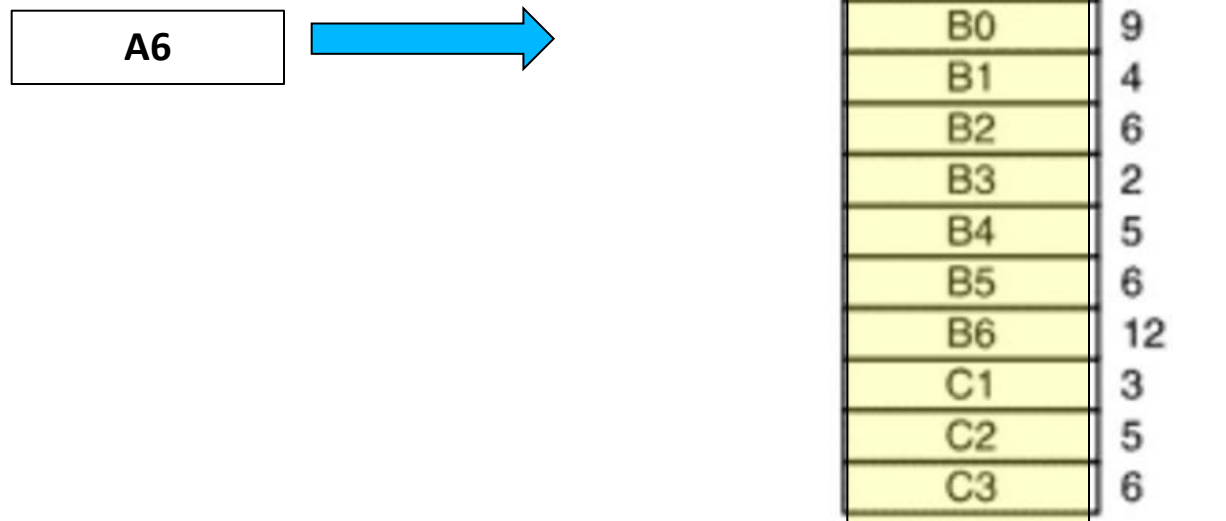


Gerenciamento de Memória

Política de substituição de páginas

Global:

- Leva em consideração os processos executáveis.

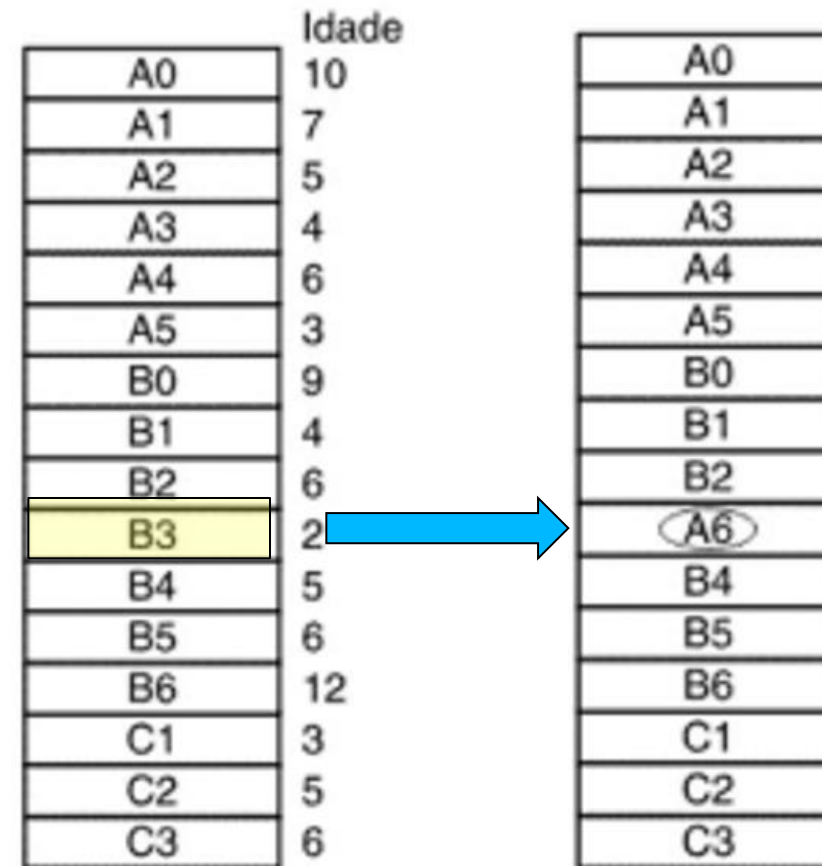


Gerenciamento de Memória

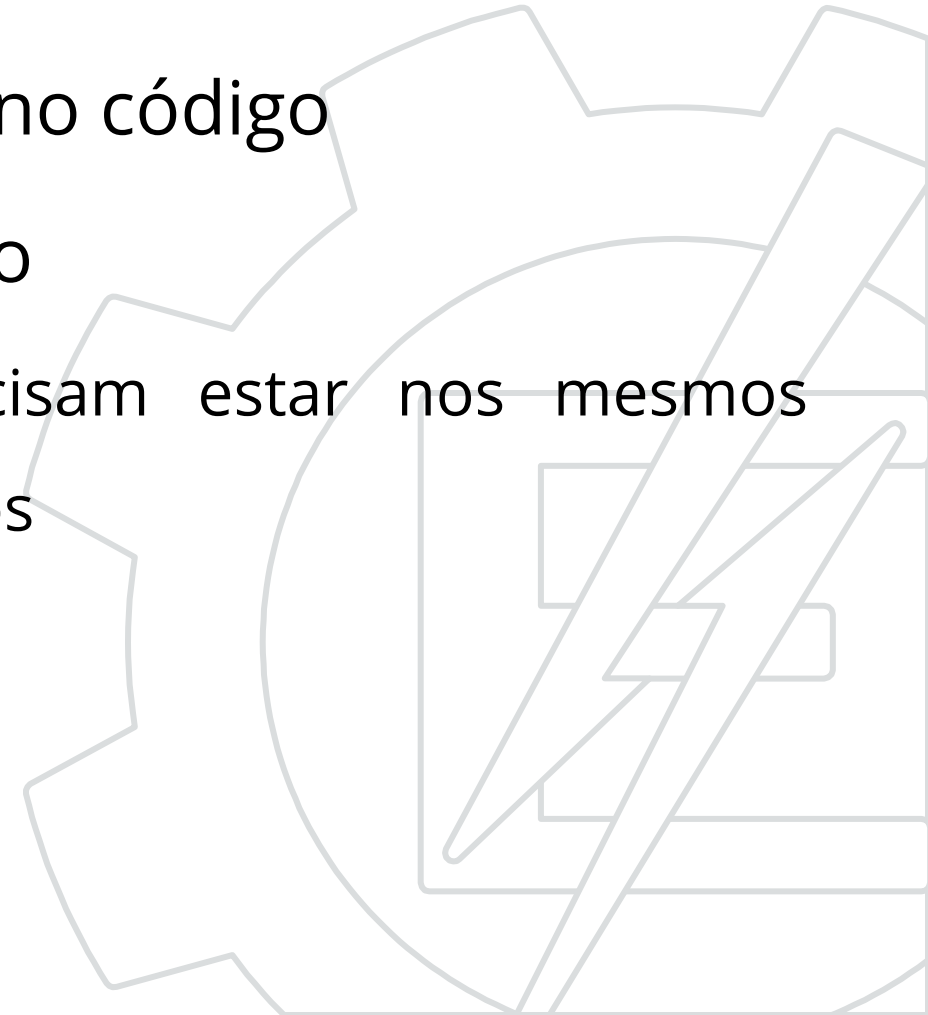
Política de substituição de páginas

Global:

- Leva em consideração os processos executáveis.



- Compartilhamento de código
 - Deve-se ter atenção com endereços no código
 - Código com endereçamento absoluto
 - Páginas com endereços usados precisam estar nos mesmos endereços lógicos em todos os processos
 - Código com endereçamento relativo
 - Não há restrições de posicionamento



Gerenciamento de Memória

Compartilhamento de páginas

ed 1
ed 2
ed 3
data 1

process P_1

3
4
6
1

page table
for P_1

ed 1
ed 2
ed 3
data 3

process P_3

3
4
6
2

page table
for P_3

ed 1
ed 2
ed 3
data 2

process P_2

3
4
6
7

page table
for P_2

0	
1	data 1
2	data 3
3	ed 1
4	ed 2
5	
6	ed 3
7	data 2
8	
9	
10	
11	



Gerenciamento de Memória

Compartilhamento de páginas

- A solução-padrão para esse problema é usar um *cache* de *hardware* especial e pequeno, de pesquisa rápida, chamado *buffer* de tradução paralelo (**TLB – *translation lookaside buffer***).
- O TLB é uma memória associativa de alta velocidade. Cada entrada do TLB é composta de duas partes: uma chave (ou *tag*) e um valor.
- Alguns TLBs armazenam identificadores do espaço de endereçamento (**ASIDs – *address-space identifiers***) em cada entrada.
- Um ASID identifica cada processo de maneira exclusiva e é usado para fornecer proteção ao espaço de endereçamento desse processo.

Gerenciamento de Memória

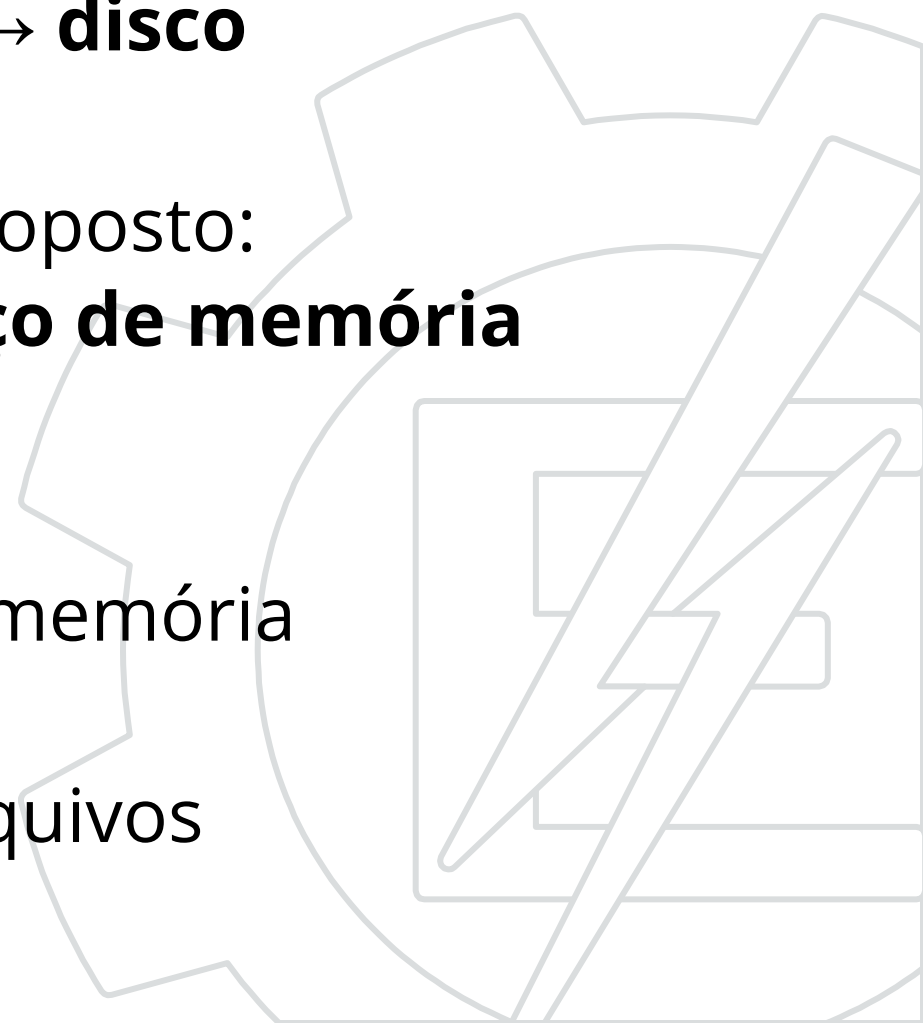
Cópia após gravação (*Copy-On-Write: COW*)

- Duplicação de página alterável sob demanda
- Processos recebem referência à mesma página
 - Página (alterável) tem permissão de escrita = 0
- Se algum dos processos tenta escrever
 - *Page fault* → nova página/quadro é alocada(o)
 - A partir daí, cada processo tem sua cópia



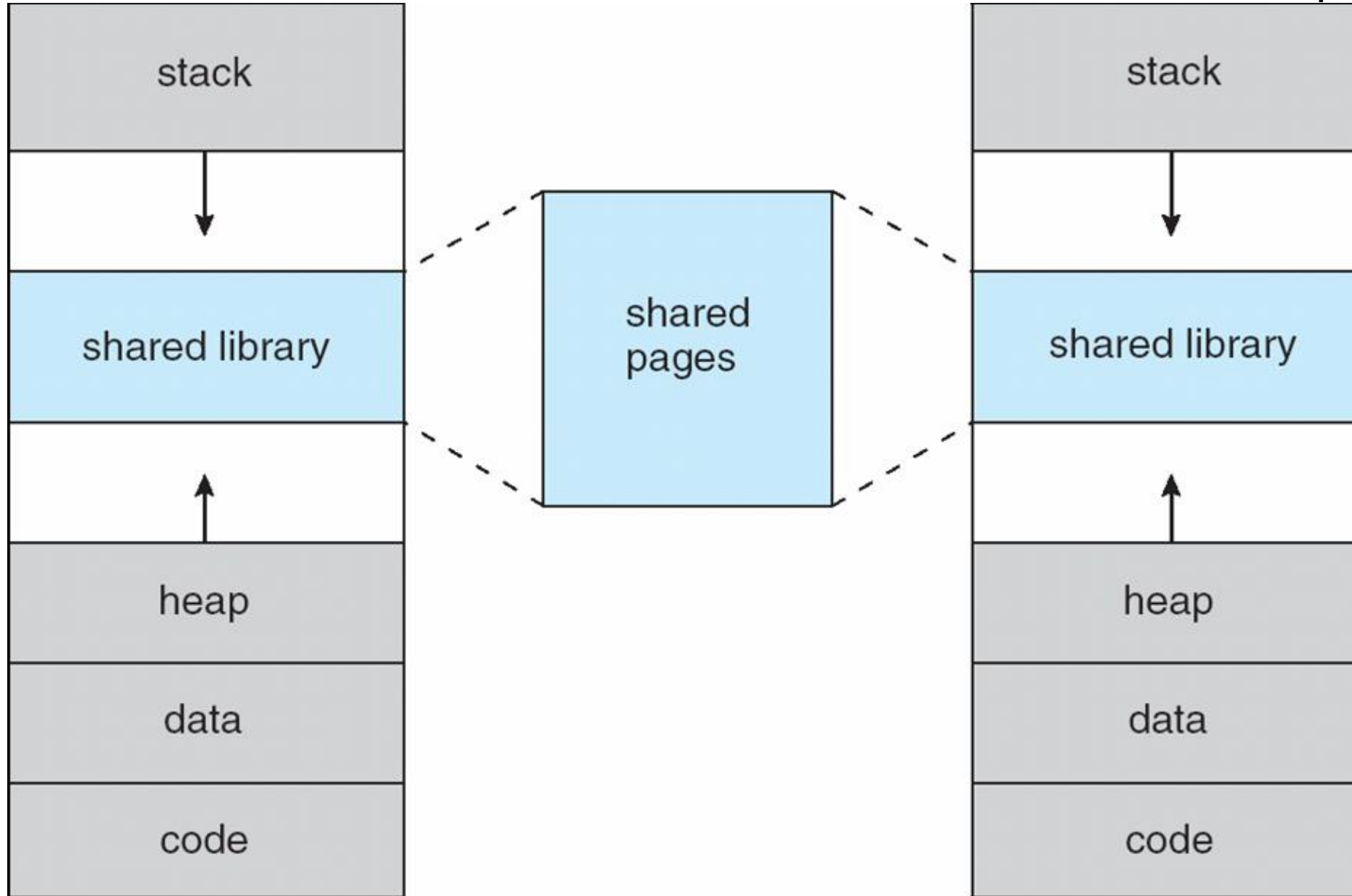
Gerenciamento de Memória

Arquivos mapeados em memória

- Memória virtual: mapeamento do endereço lógico do processo a uma área em disco
 - Normalmente, o sentido é **memória** → **disco**
 - No Unix, com o `mmap()`, pode-se fazer o oposto:
 - Conteúdo do disco (**arquivo**) → **espaço de memória**
 - Simplifica o acesso a arquivos de dados
 - Podem ser tratados como vetores na memória
 - Permite que processos compartilhem arquivos
- 

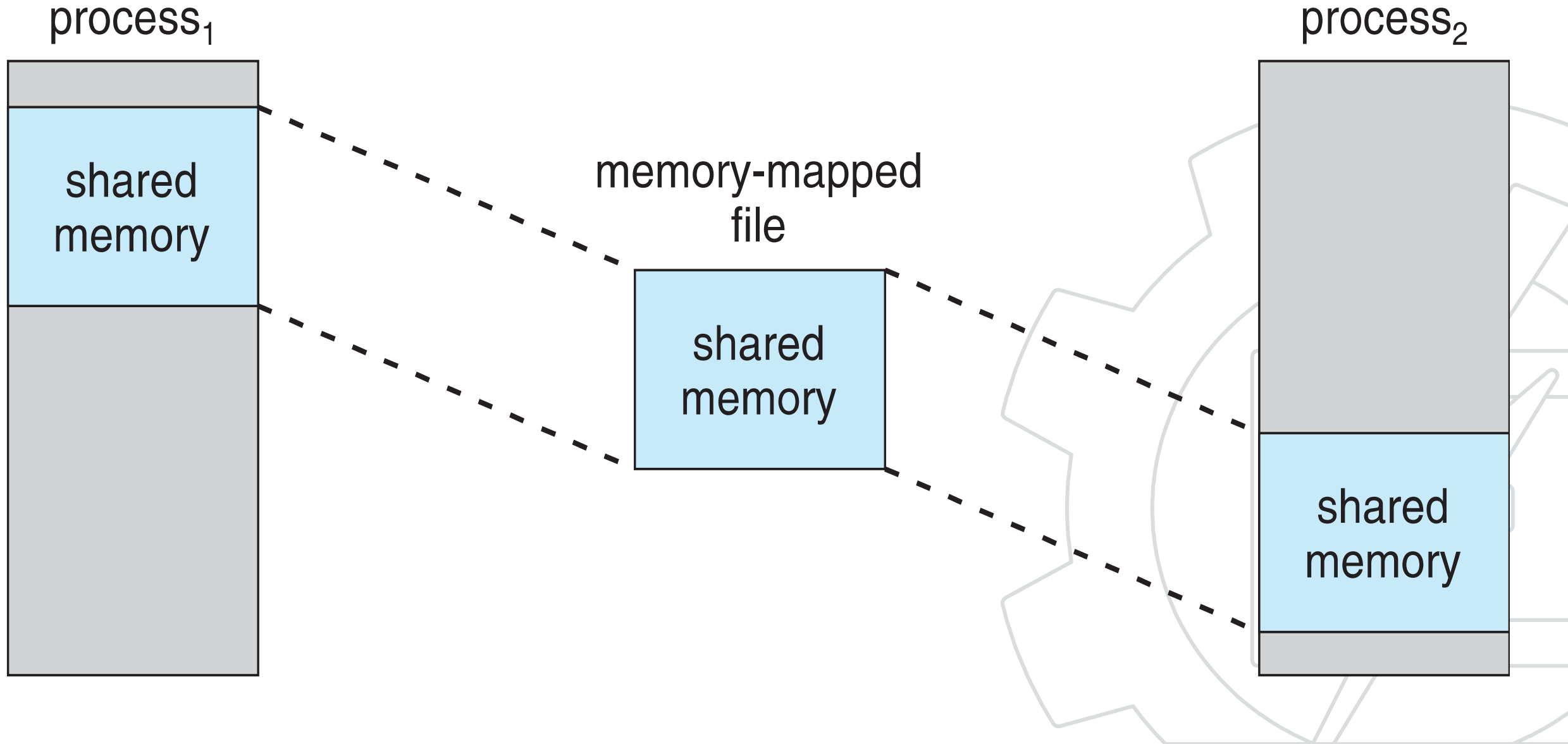
Gerenciamento de Memória

Arquivos mapeados em memória



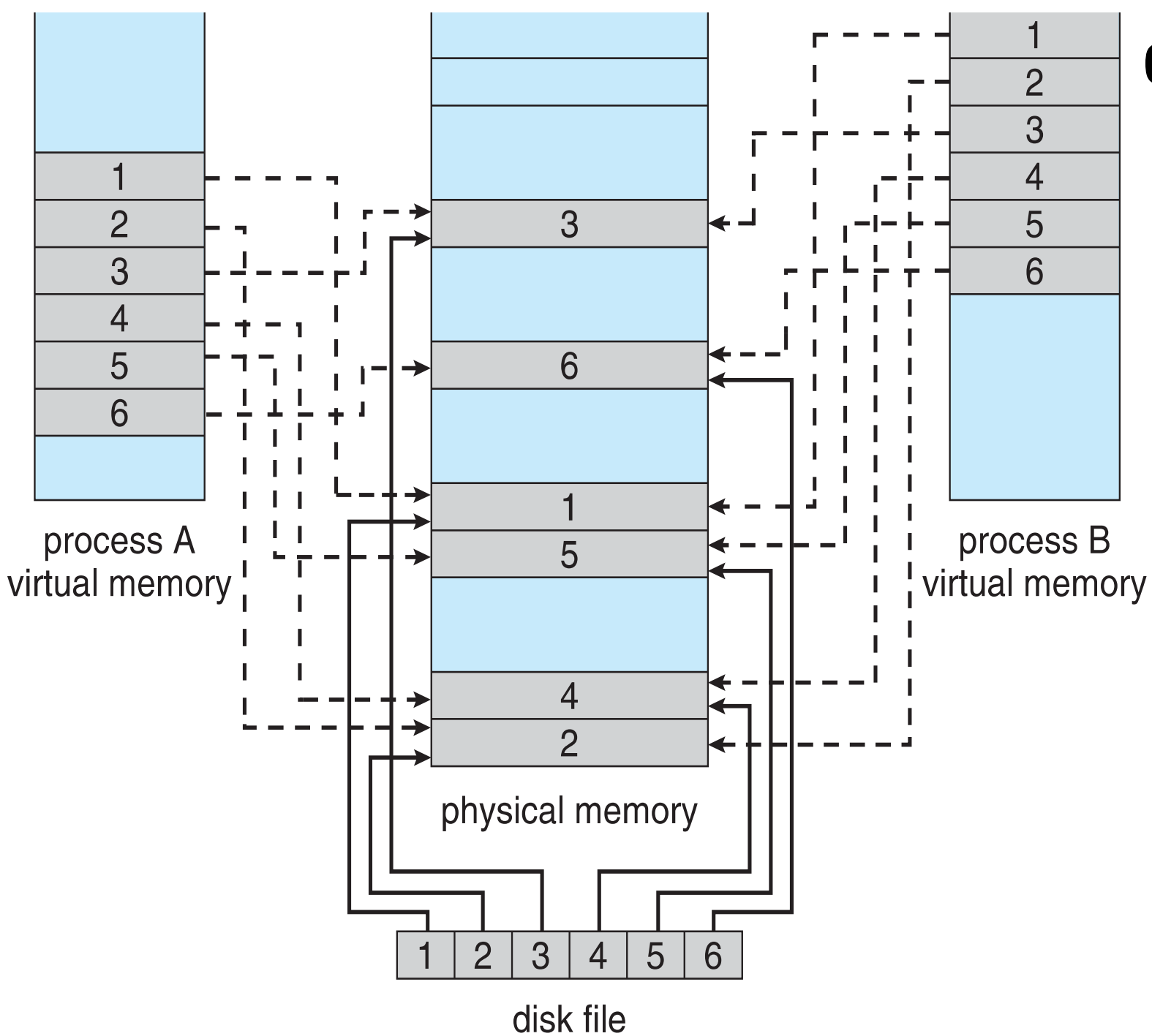
Gerenciamento de Memória

Arquivos mapeados em memória



Gerenciamento de Memória

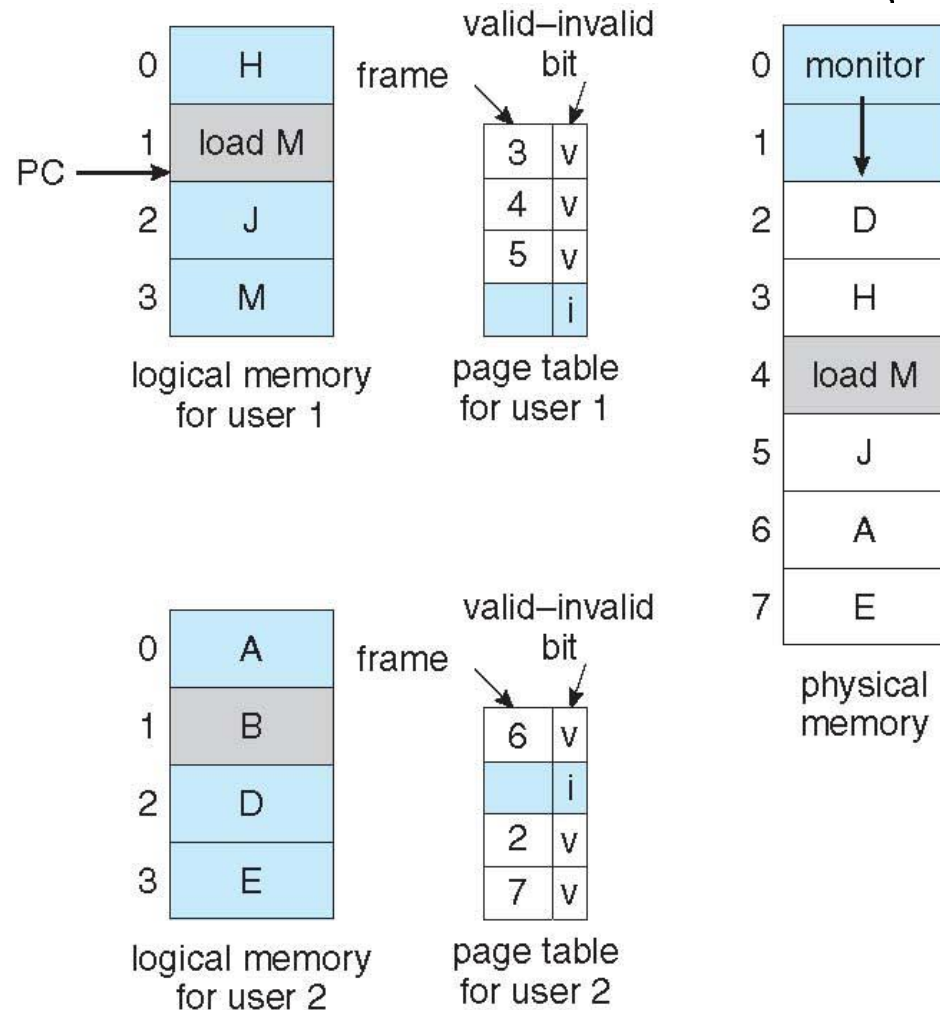
Arquivos mapeados em memória



Gerenciamento de Memória

Superlocação de memória

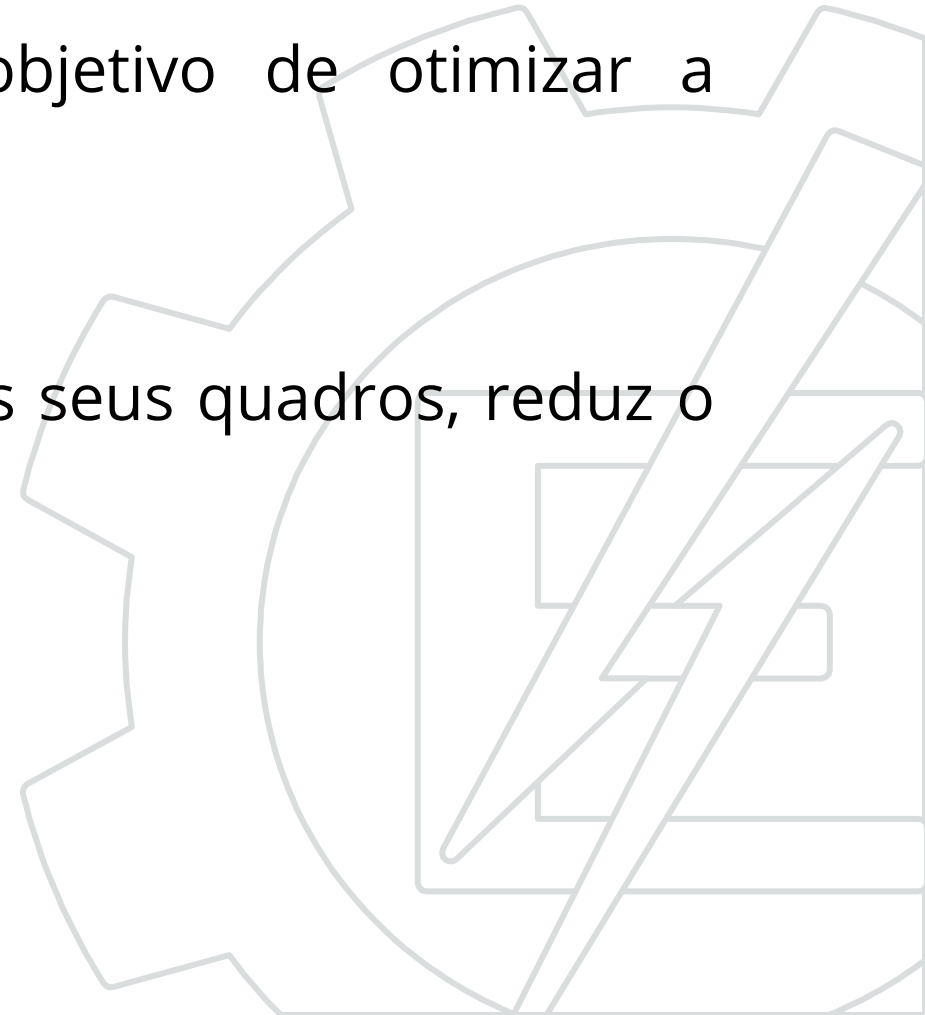
- Processo em execução gera erro de página → S.O. determina onde ela reside no disco
 - E se não houver um quadro vazio? (toda mem. em uso)
 - Há necessidade de substituição de páginas



Gerenciamento de Memória

Superlocação de memória

- Quando não houver um quadro vazio, o S.O. pode:
 - Encerrar o processo do usuário
 - **Péssima escolha**, pois vai contra o objetivo de otimizar a utilização
 - Remover um processo da memória
 - **Boa escolha**, mas como remove todos os seus quadros, reduz o nível de multiprogramação
 - Realizar a substituição de páginas
 - **Solução mais comum**

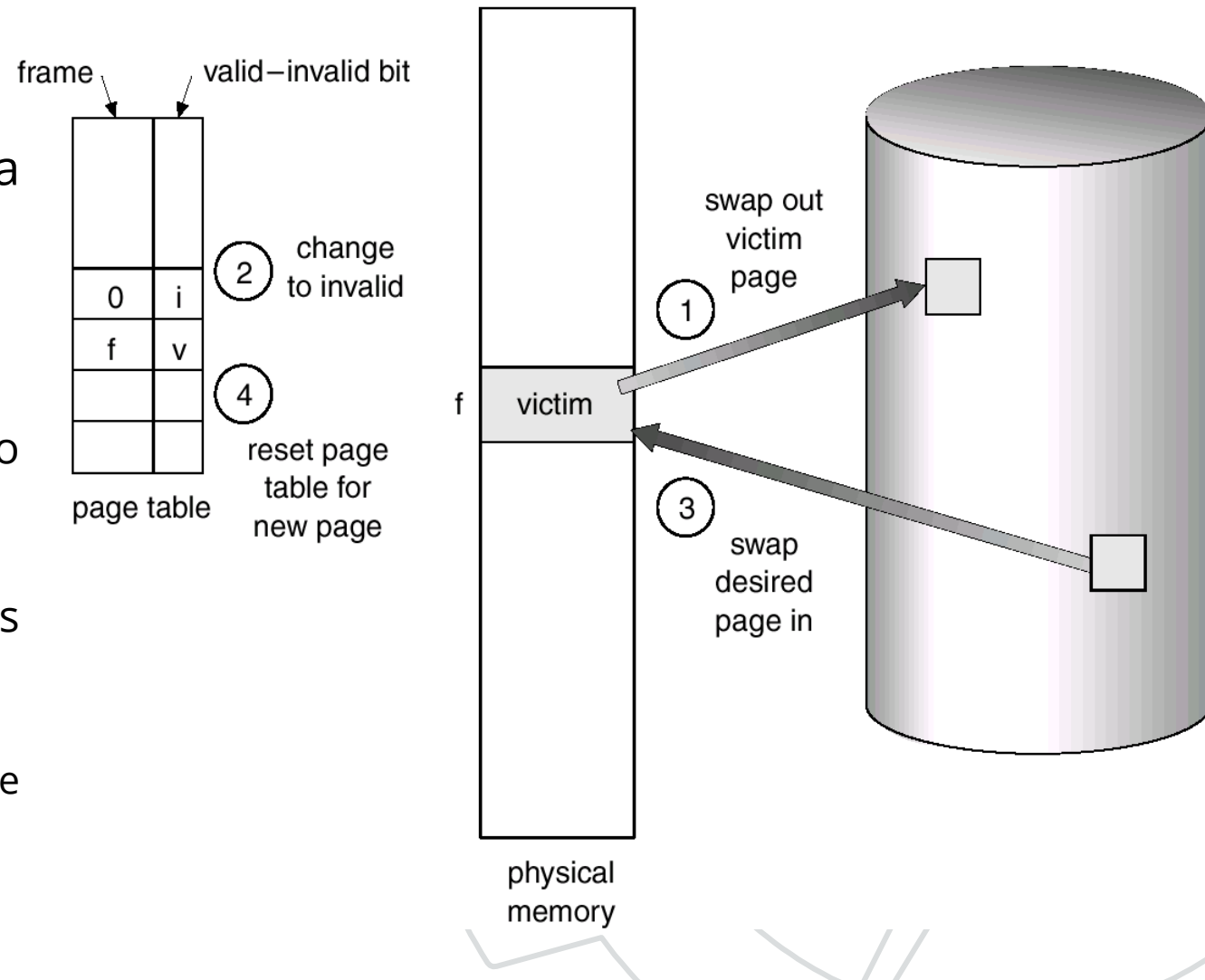


Gerenciamento de Memória

Substituição de páginas

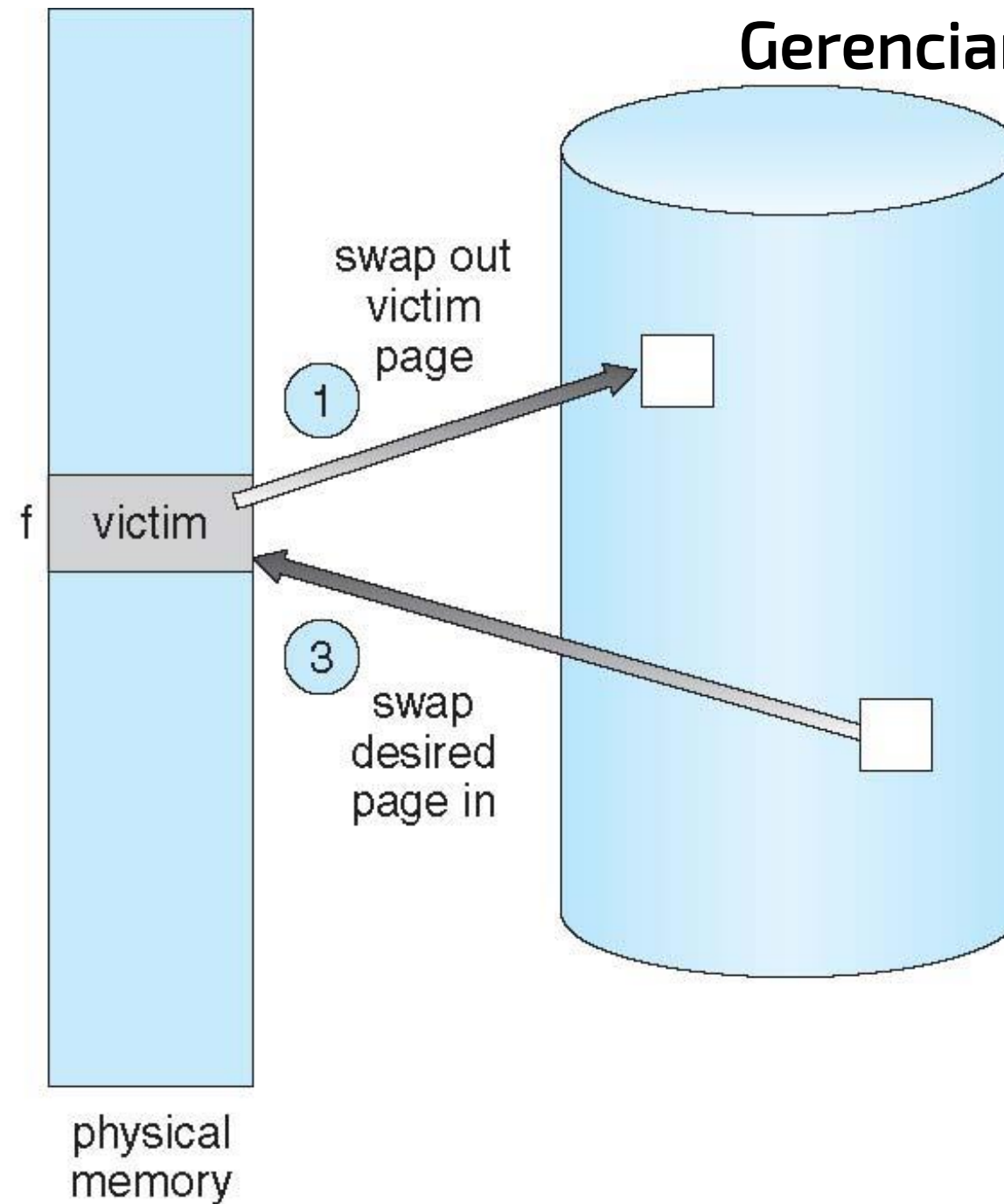
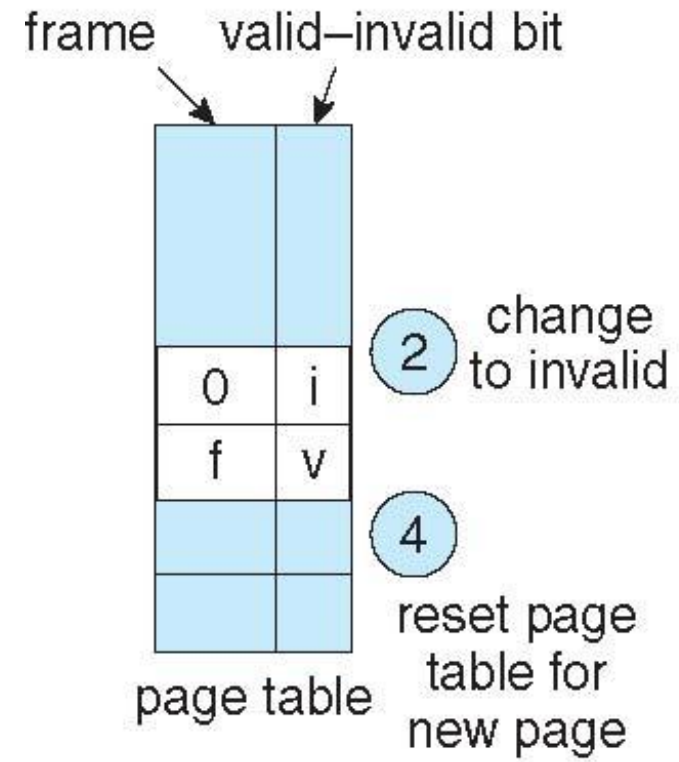
Substituição/reposição de páginas:

- Se nenhum quadro estiver livre
- Encontre uma página que não esteja em uso
- Libere o quadro dela
 - Grave o conteúdo do quadro no espaço de permuta
 - Modifique a tabela de páginas invalidando a página vítima
- OBS.: a página retirada pode vir a ser acessada de novo.



Gerenciamento de Memória

Substituição de páginas

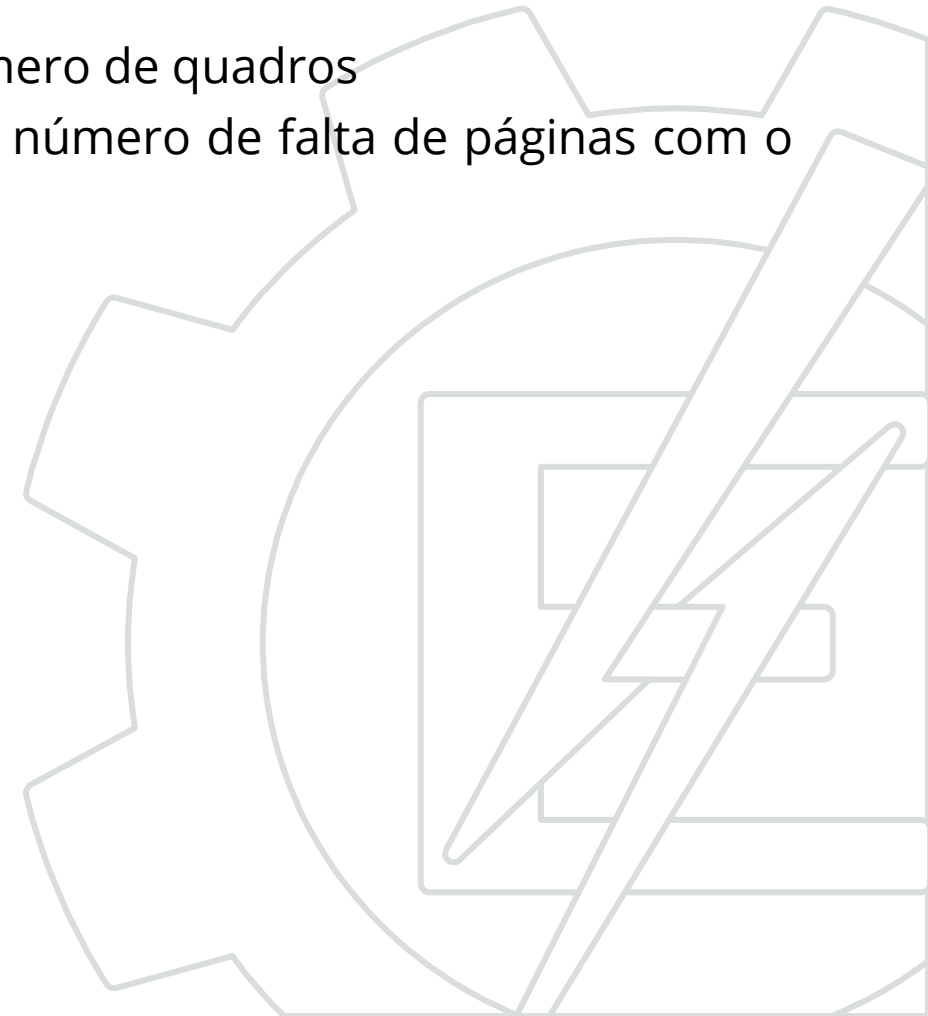
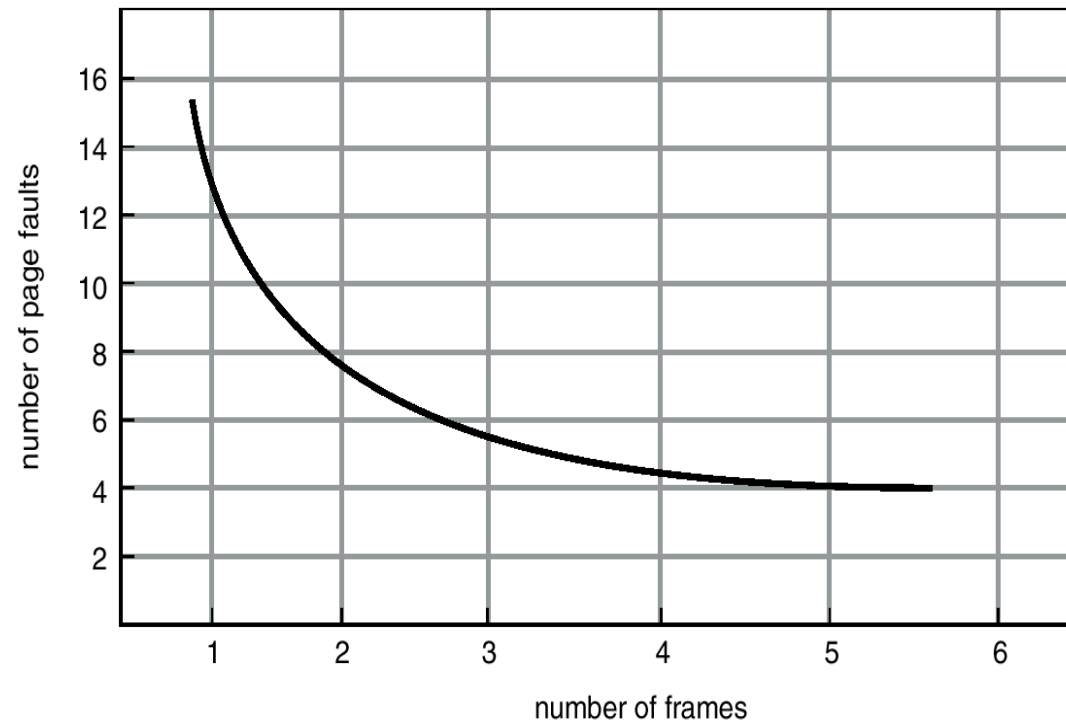


- Melhoria na seleção da página vítima
 - Tabela de páginas inclui bit extra para auxiliar no processo de escolha de candidatos
 - **Bit de modificação** (ou *bit poluído* ou *bit sujo*)
 - **Bit ligado**: página foi modificada desde que foi lida no disco, devemos gravar a página no disco, custo maior
 - **Bit desligado**: página não modificada, não precisa gravar no disco, pode ser descartada a um custo menor
- Substituição de páginas é essencial para a paginação por demanda
 - Completa a separação entre memória física e lógica
 - Tamanho do espaço de end. lógico não é mais limitado pela mem. física
 - Mem. virtual enorme oferecida ao programador numa mem. física menor

Gerenciamento de Memória

Relação entre falhas e nº de quadros disponíveis

- Existem muitos algoritmos de substituição de páginas nos S.O.
 - Prefere-se aquele com a taxa de erros de página mais baixa
 - Em geral, com a curva abaixo: + quadros \rightarrow - erros
 - Obs.1: aumentando a memória física (RAM), aumenta-se o número de quadros
 - Obs.2: Espera-se (modelo ideal) que este gráfico represente o número de falta de páginas com o aumento no número de quadros:

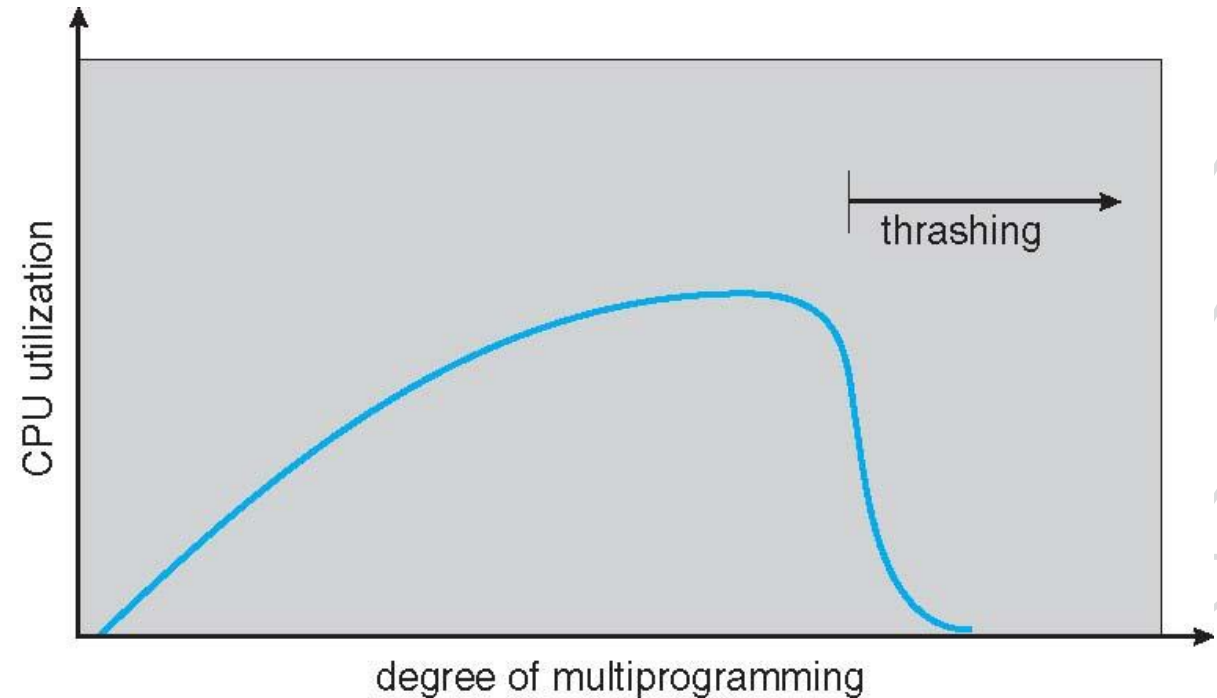


Gerenciamento de Memória

Trashing

- Se processos não têm páginas “suficientes”

- *Erros de página* aumentam
- Utilização de CPU diminui
- S.O. precisa aumentar a multiprog.
- Outro proc. é trazido para a mem.
- Demanda por quadros aumenta
- *Erros de página* aumentam



- **Thrashing:** processos estão ocupados apenas fazendo *swap* de páginas e aguardando na fila do paginador.

Algoritmos de substituição de páginas



Gerenciamento de Memória

Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Gerenciamento de Memória

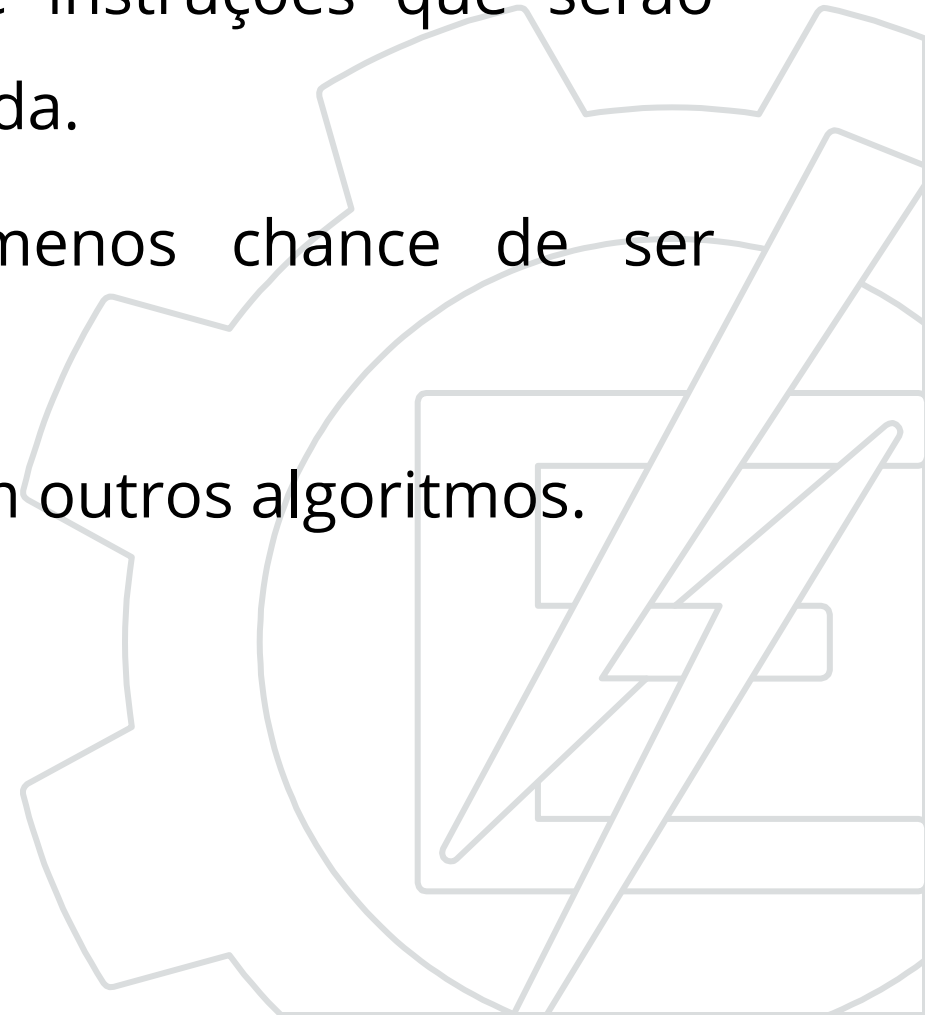
Algoritmos de substituição de páginas

- **Ótimo**
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Algoritmo Ótimo

- Cada página é marcada com o número de instruções que serão executadas antes que a página seja referenciada.
- Retira da memória a página que tem menos chance de ser referenciada.
- Utilizado em simulações para comparação com outros algoritmos.
- Praticamente impossível de se implementar.

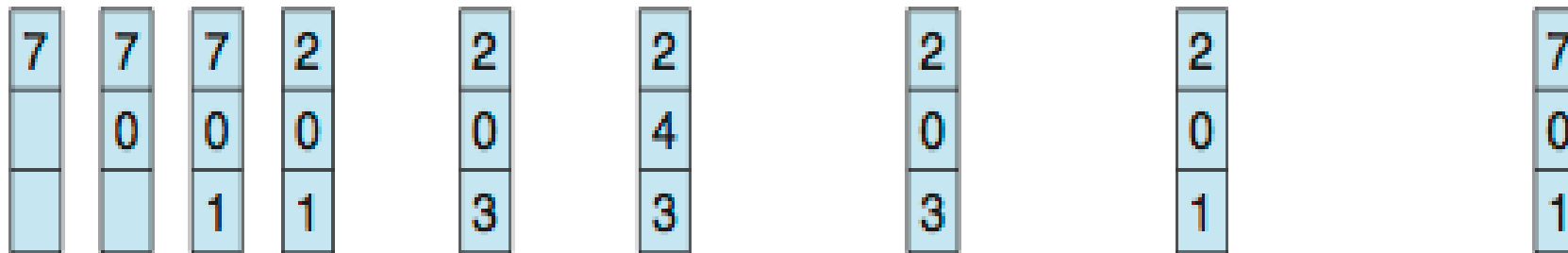


Algoritmo Ótimo

- Utiliza o menor tempo possível a partir do conhecimento do momento em que a página será referenciada e o exato momento da substituição.

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

Gerenciamento de Memória

Algoritmos de substituição de páginas

- Ótimo
- **NRU (*Not-Recently Used*)**
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Algoritmo *Not-Recently Used*

- O Sistema Operacional coletar estatísticas de uso da página.
- São fornecidos dois bits à página: **R**referenciada e **M**odificada
 - Classe 0 (00) → Não-referenciada e não modificada
 - Classe 1 (01) → Não-referenciada e modificada
 - Classe 2 (10) → Referenciada e não modificada
 - Classe 3 (11) → Referenciada e modificada
- **R**referenciada significa lida ou escrita
- **M**odificada significa escrita



Algoritmo *Not-Recently Used*

- R e M são atualizados a cada referência à memória.
- R e M são armazenados em cada entrada da tabela de páginas
- No início, R e M são iguais a zero para todas as páginas
- **Periodicamente o bit R é limpo**, com o objetivo de permitir a marcação somente das páginas que foram referenciadas recentemente → A cada *tick* do relógio.
- **O bit M não é limpo**, pois o S.O. precisa saber se deve escrever a página no disco.

Gerenciamento de Memória

Algoritmos de substituição de páginas

Algoritmo *Not-Recently Used*

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		4	4	4	0			1		1		1
	0	0	0		0		0	0	3	3			3		0		0
		1	1		3		3	2	2	2			2		2		7

page frames

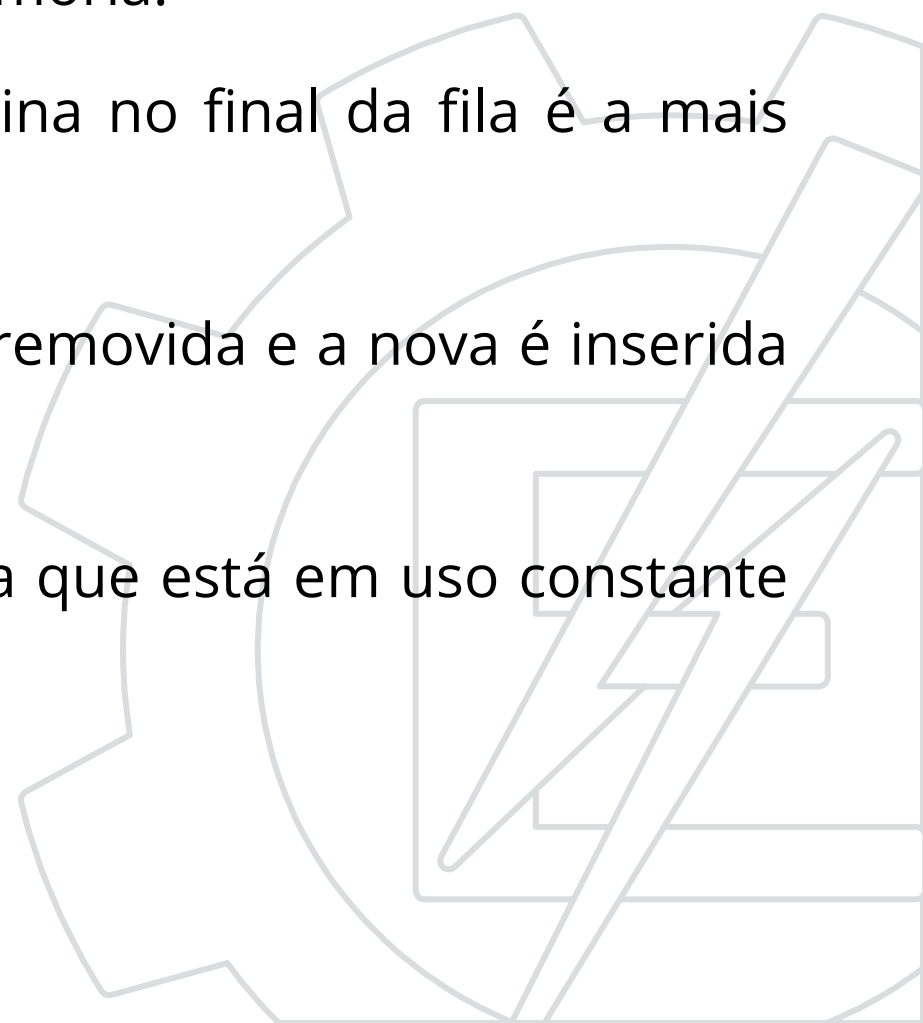
Gerenciamento de Memória

Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- **FIFO (*First-In First-Out*)**
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Algoritmo FIFO (*First-In First-Out*)

- S.O. mantém uma lista de páginas correntes na memória.
 - A página no início da fila é a mais antiga e a página no final da fila é a mais nova.
 - Quando ocorre um *page fault*, a página do início é removida e a nova é inserida no final da fila.
 - Simples, mas pode ser ineficiente, pois uma página que está em uso constante pode ser retirada.
 - É pouco utilizado neste contexto.
- 

Gerenciamento de Memória

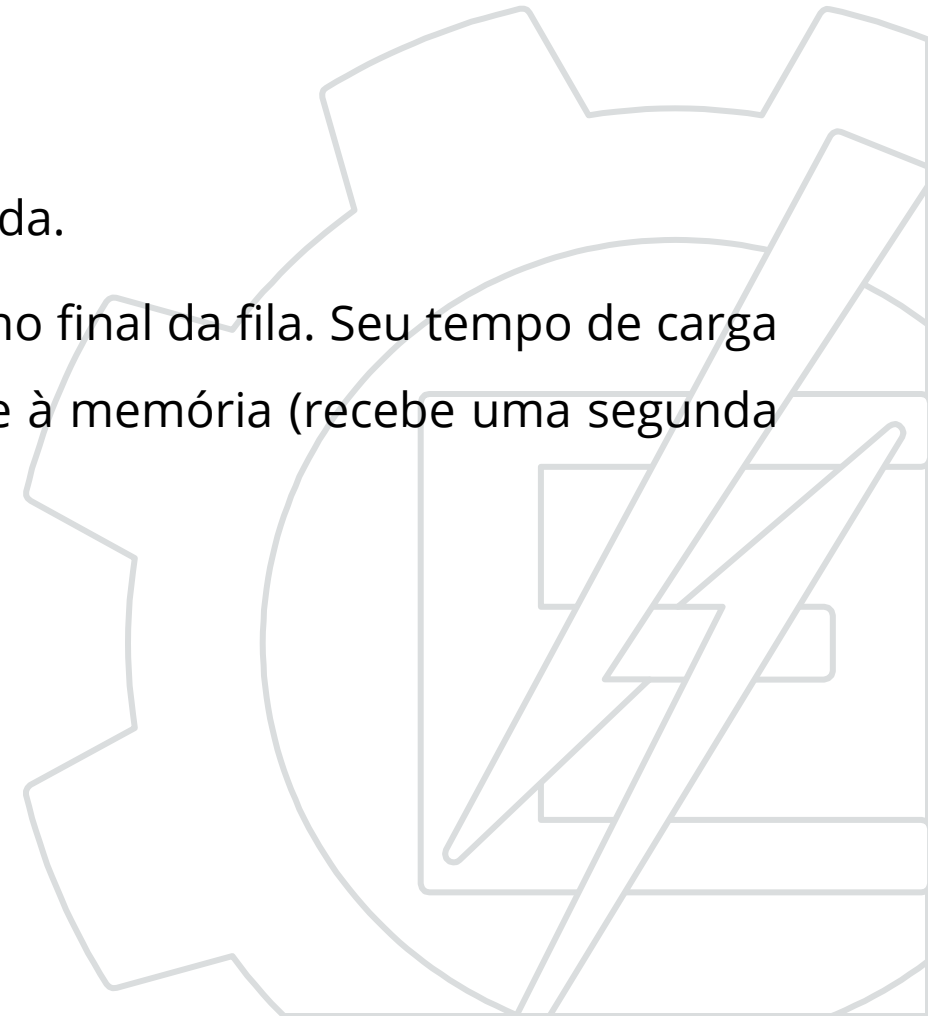
Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- **Segunda chance**
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Algoritmo Segunda Chance

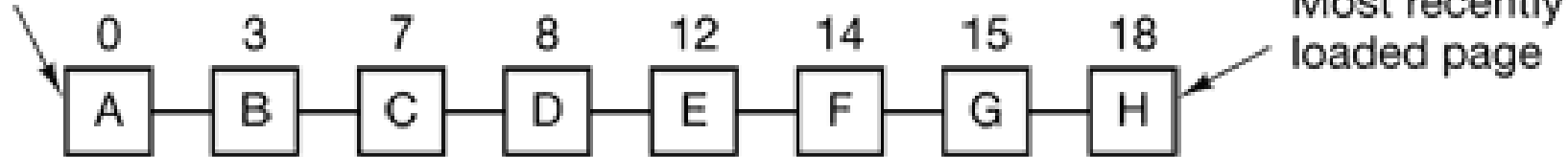
- FIFO + bit R
- Inspecciona o bit R da página mais velha
 - Se for 0, é velha e não usada recentemente, então é trocada.
 - Se for 1, o bit R é convertido em 0 e a página é colocada no final da fila. Seu tempo de carga é modificado, fazendo parecer que chegou recentemente à memória (recebe uma segunda chance).
- A busca continua pelo processo vítima.



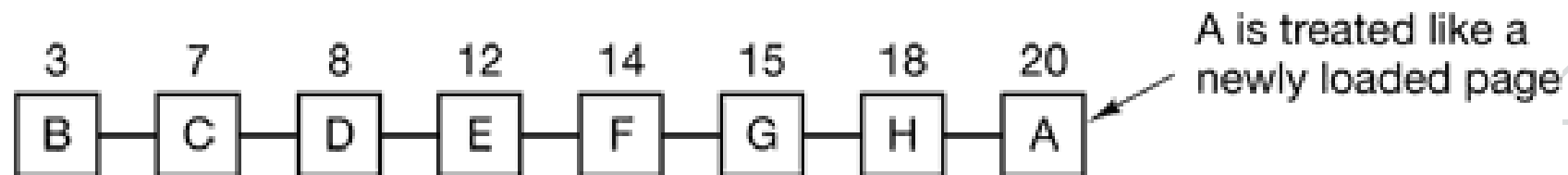
Algoritmo Segunda Chance

- Ocorre *page fault* no tempo 20 e $R_A=0$, então A é removido e o novo elemento é inserido no final.
- Ocorre *page fault* no tempo 20 e $R_A=1$, então:

Page loaded first



(a)

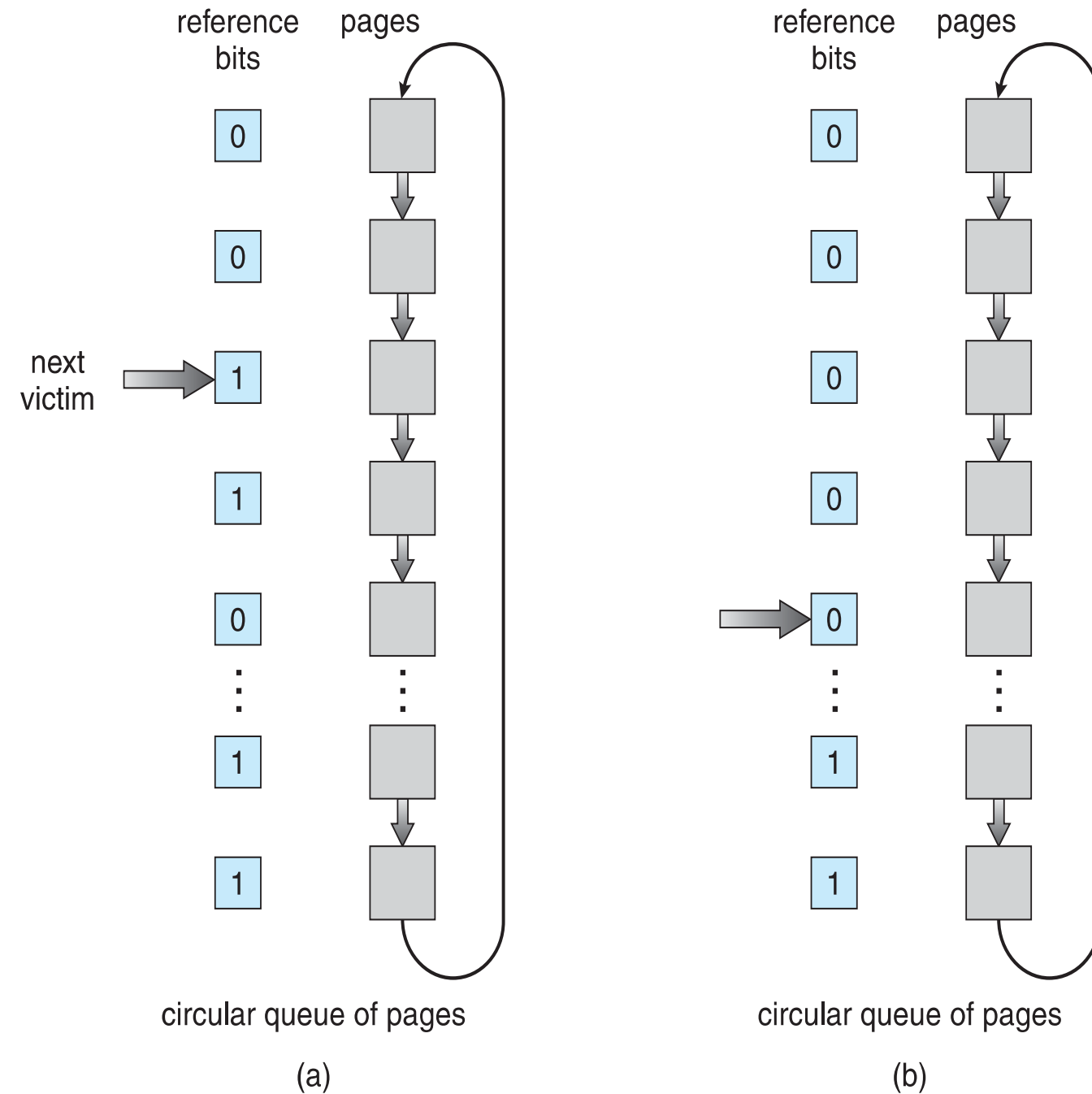


(b)

Gerenciamento de Memória

Algoritmos de substituição de páginas

Algoritmo Segunda Chance



Gerenciamento de Memória

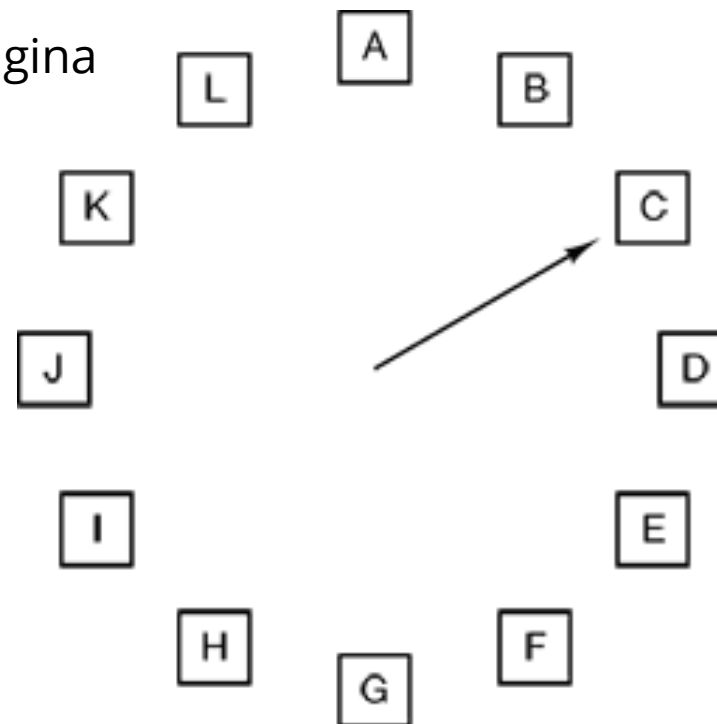
Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- **Relógio (Clock)**
- LRU (*Least-Recently Used*)
- *Working Set*
- *WSClock*



Algoritmo do Relógio (*Clock*)

- Melhoria para o algoritmo Segunda Chance
- Lista circular com ponteiro apontando para a página mais antiga, na forma de um relógio.
 - Se o bit R é igual a zero, substitui a página
 - Se não, avança para o próximo.



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

Gerenciamento de Memória

Algoritmos de substituição de páginas

Algoritmo do Relógio (*Clock*)

- Lista circular com ponteiro apontando para a página mais antiga, na forma de um relógio.
- Se o bit R é igual a zero, substitui a página
- Se não, avança para o próximo.

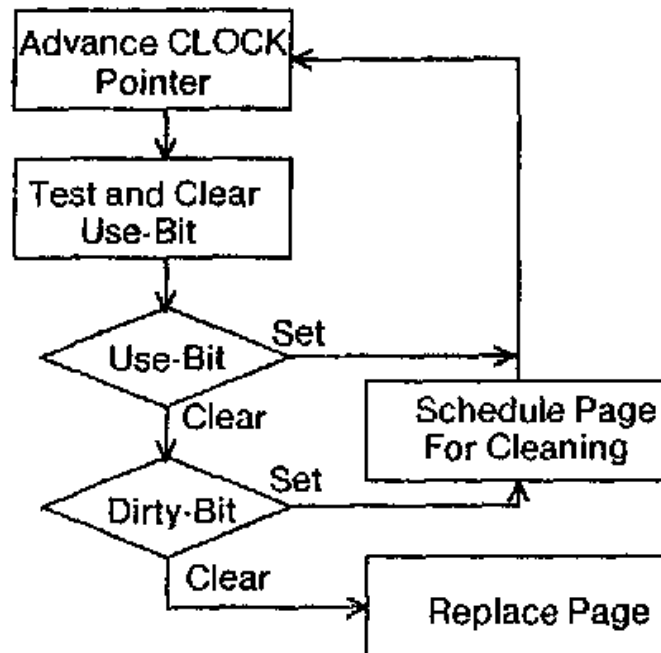
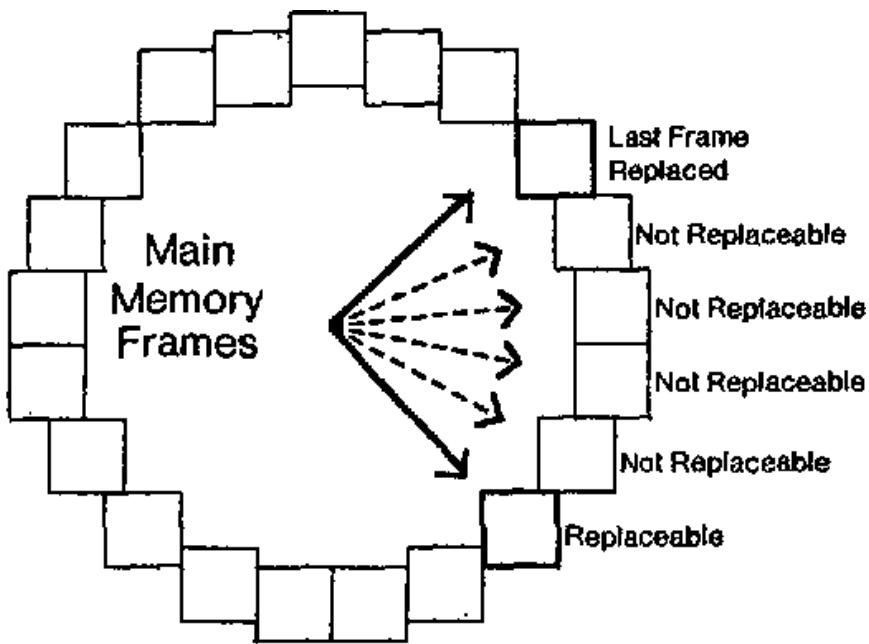


Figure 1. CLOCK Replacement Algorithm

Gerenciamento de Memória

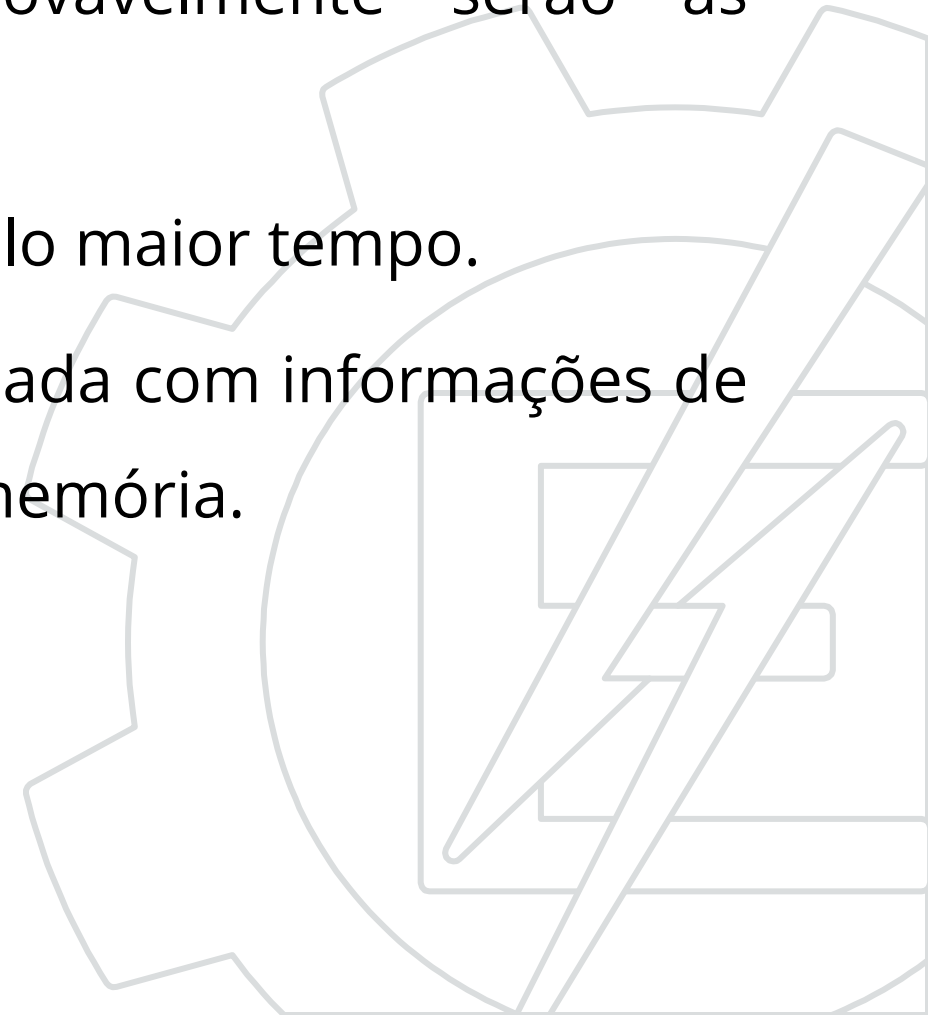
Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- **LRU (*Least-Recently Used*)**
- *Working Set*
- *WSClock*



Algoritmo *Least Recently Used* (LRU)

- Páginas muito utilizadas ultimamente provavelmente serão as próximas a serem utilizadas.
- Troca a página que permaneceu em desuso pelo maior tempo.
- Alto custo: Deve ser mantida uma lista encadeada com informações de todas as páginas mais recentes que estão na memória.
- Implementada em *hardware* ou *software*.



reference string

4 7 0 7 1 0 1 2 1 2 7 1 2

2
1
0
7
4

stack
before
a

7
2
1
0
4

stack
after
b

↑
a

↑
b

Gerenciamento de Memória

Algoritmos de substituição de páginas

Algoritmo *Least Recently Used* (LRU)

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7

7
0

7
0
1

2
0
1

2
0
3

4
0
3

4
0
2

4
3
2

0
3
2

1
3
2

1
0
2

1
0
7

page frames

LRU em *hardware*

- MMU deve suportar a implementação LRU.
- Contador em *hardware* (64 bits) incrementado automaticamente após cada acesso.
- Tabela de páginas armazena o valor desse contador em cada tarefa.
 - Em um *page fault*, o S.O. examina todas as entradas na tabela para encontrar o menor valor do contador.

LRU em *software* (NFU - *Not Frequently Used*)

- Para cada página existe um contador implementado em *software*, iniciado em zero.
- Em um *page fault*, o S.O. escolhe a página com o menor contador.
- Problema:
 - Esse algoritmo não se esquece de nada → Páginas frequentemente acessadas (em grande quantidade) no início não serão candidatas a vítima, mesmo que não estejam em uso.
 - Utilização de envelhecimento (*aging*).

Gerenciamento de Memória

Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- **Working Set**
- *WSClock*



Algoritmo *Working Set*

- Conjunto de páginas que um processo está efetivamente utilizando em um determinado tempo t .
- Um processo só é executado quando todas as páginas necessárias no tempo t estão na memória.
- A ideia é determinar o *working set* de cada processo e tê-lo na memória antes de rodar o processo.

Algoritmo *Working Set*

$\Delta \equiv \text{working-set window} \equiv$ um número fixo de referenciamentos de página, por exemplo, 10.000 instruções

WSS_i (*working set* do Processo P_i) = número total de páginas referenciadas no intervalo mais recente Δ

- Se Δ for muito pequeno não abrangerá toda a localidade
- Se Δ for muito grande abrangerá várias localidades
- Se Δ tende ao infinito abrangerá todo o programa

Algoritmo *Working Set*



Working sets of this process at these time instants will be:

$$WS(t_1) = \{2, 1, 5, 7\}$$

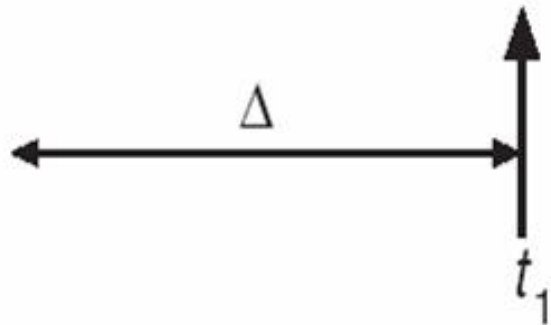
$$WS(t_2) = \{7, 5, 1, 3, 4\}$$

$$WS(t_3) = \{3, 4\}$$

Algoritmo *Working Set*

page reference table

... 2 6 1 5 7 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 1 3 2 3 4 4 4 3 4 4 4 ...



$$WS(t_1) = \{1, 2, 5, 6, 7\}$$



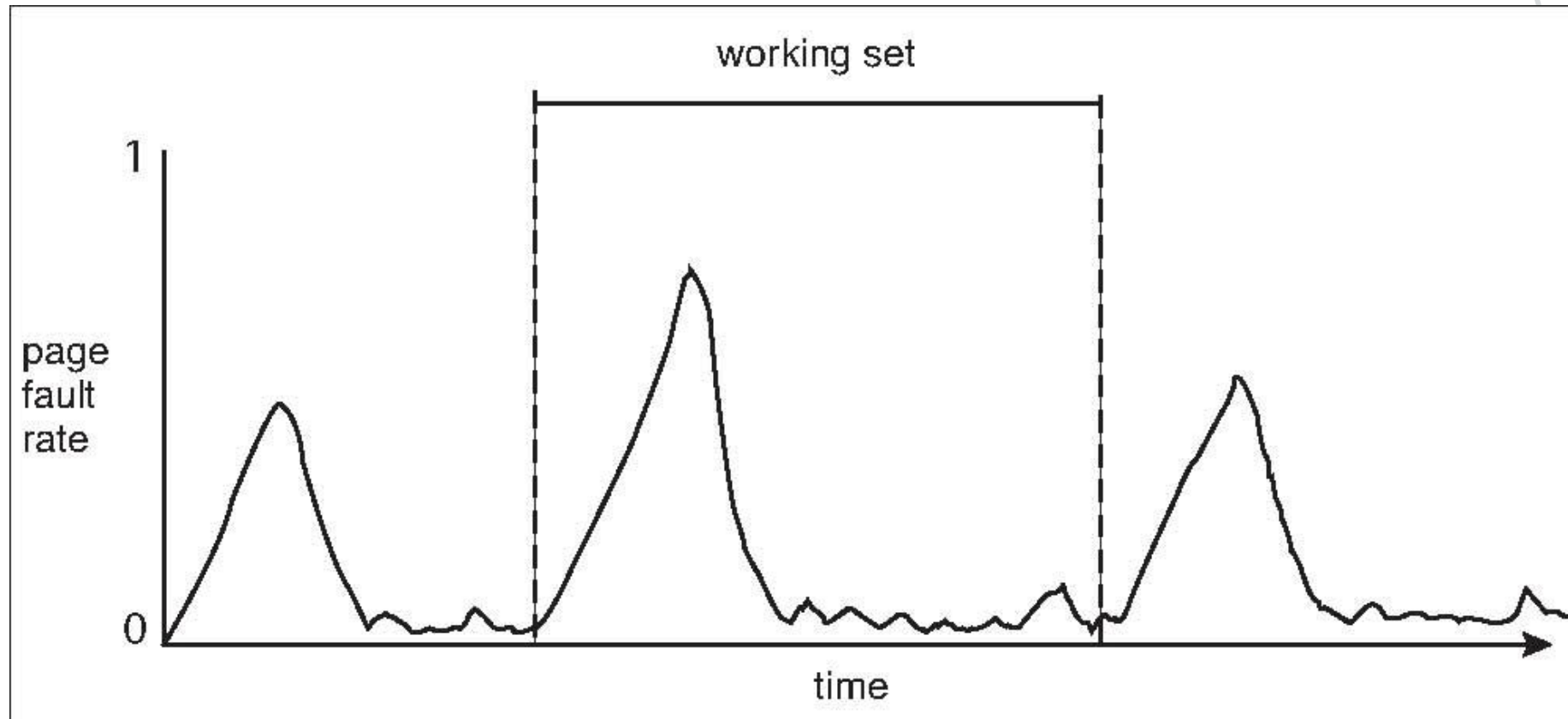
$$WS(t_2) = \{3, 4\}$$

Algoritmo *Working Set*

- Podemos estimar o número de páginas necessárias quando o programa é trazido do disco com base em seu *working set* de quando foi interrompido.
- **Pré-paginação** consiste em carregar estas páginas antes de rodar novamente o processo.
- O *working set* pode ser visto como o conjunto de páginas que o programa referenciou durante os últimos t segundos de sua execução.
- Utiliza o bit **R**.

Algoritmo *Working Set*

- O conjunto de trabalho depende da quantidade de páginas referenciadas e varia durante a execução:

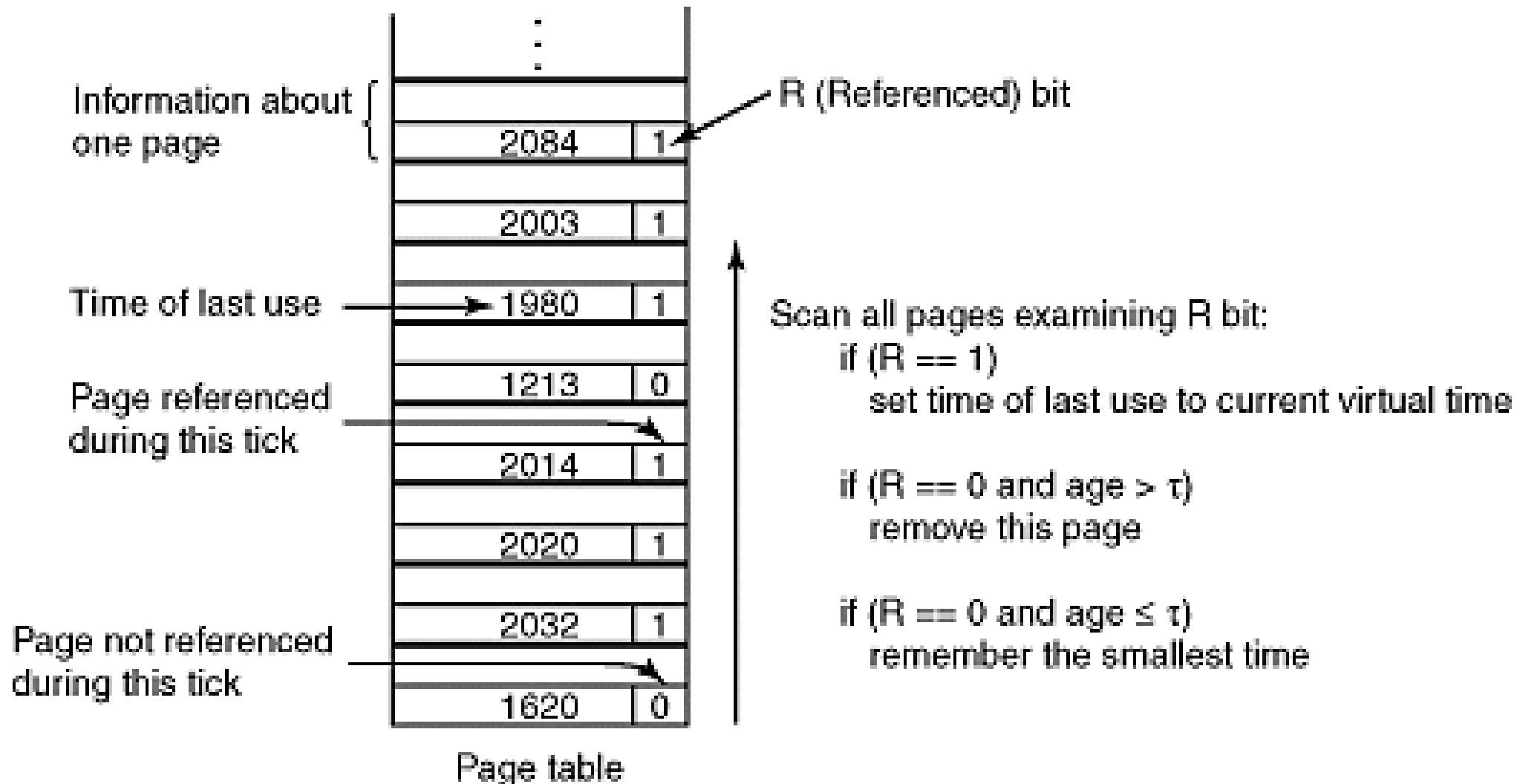


Gerenciamento de Memória

Algoritmos de substituição de páginas

Algoritmo *Working Set*

2204 Current virtual time



Gerenciamento de Memória

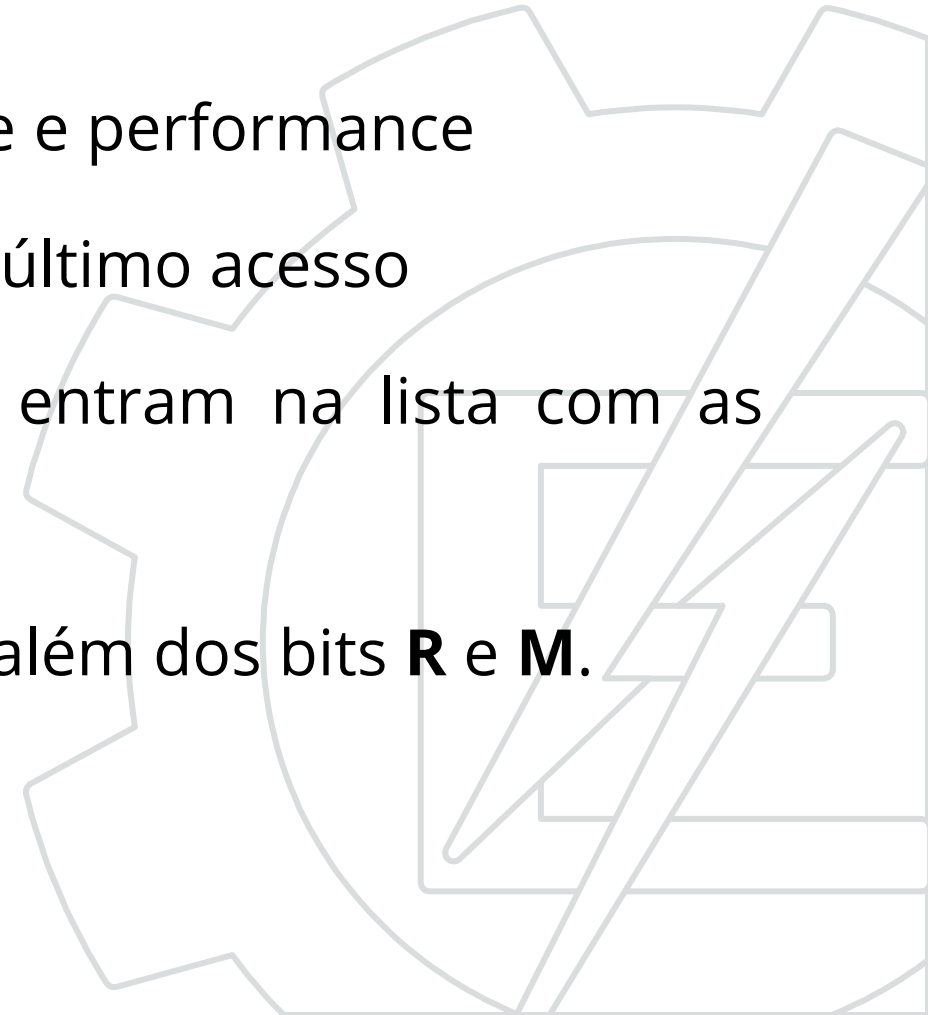
Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio
- LRU (*Least-Recently Used*)
- *Working Set*
- **WSClock**

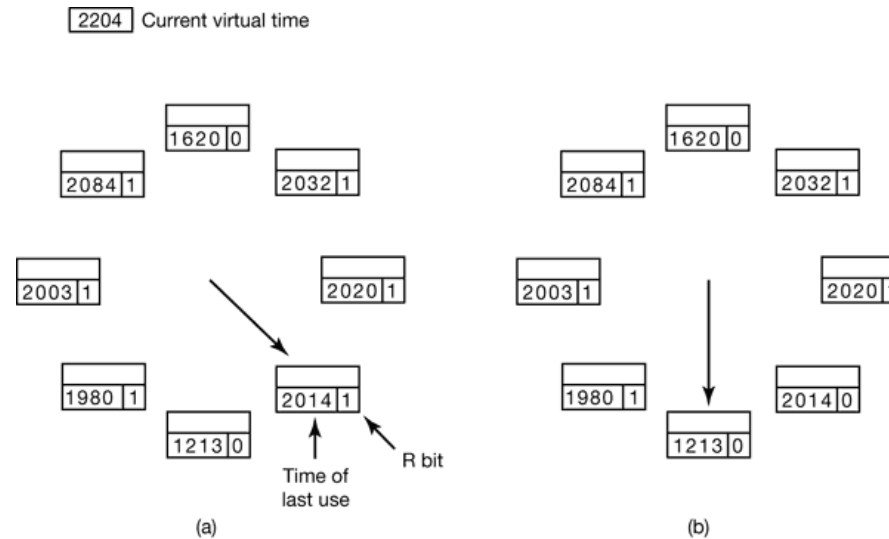


Algoritmo *Working Set Clock* (WSClock)

- *Clock + Working Set*
- Amplamente utilizado devido à sua simplicidade e performance
- Utiliza lista circular de páginas com o tempo do último acesso
- À medida que mais páginas são carregadas, entram na lista com as páginas do *working set*.
- Cada entrada contém o **tempo do último uso**, além dos bits **R** e **M**.



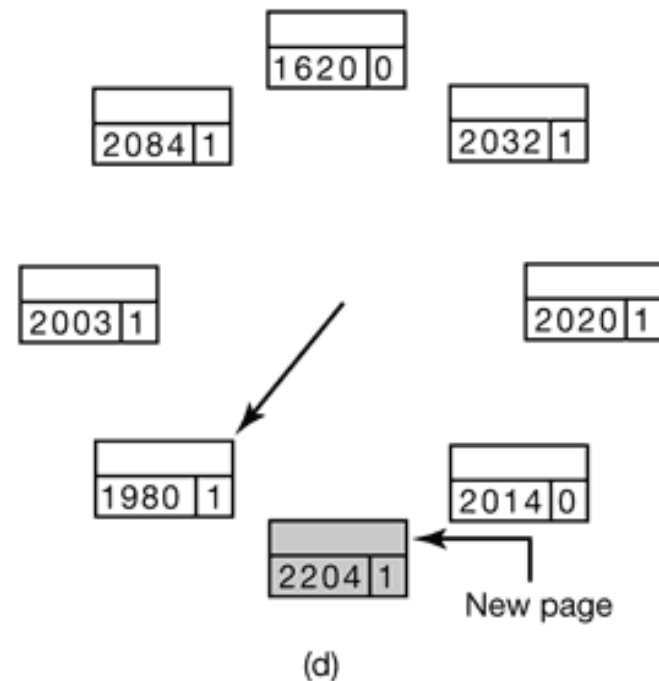
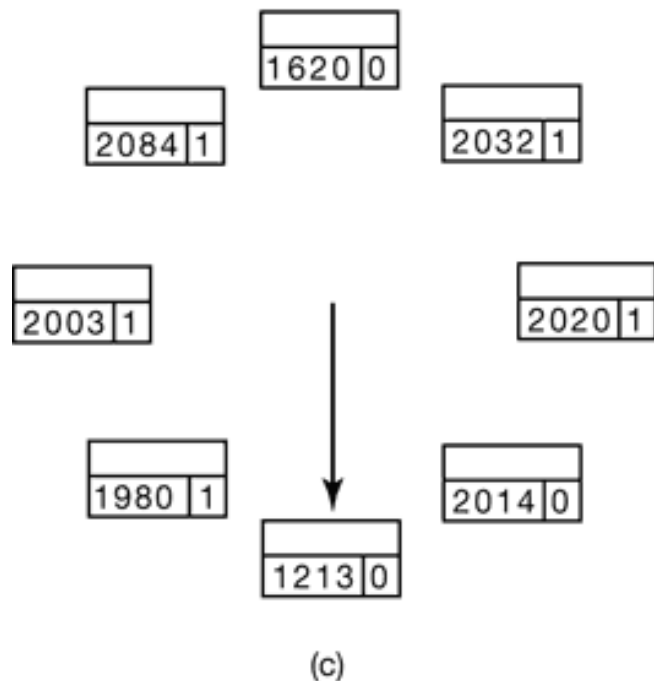
Algoritmo *Working Set Clock* (WSClock)



- A cada *page fault*, a página apontada é examinada através do bit **R**.
 - Se o bit for igual a 1, a página foi utilizada durante o último ciclo de *clock*.
 - O bit é zerado e a próxima página é analisada.

Algoritmo *Working Set Clock (WSClock)*

- A cada *page fault*, a página apontada é examinada através do bit **R**.
 - Se o bit for igual a 0, a página não foi utilizada durante o último ciclo de *clock*, então a página é substituída.



Gerenciamento de Memória

Algoritmos de substituição de páginas

- Ótimo
- NRU (*Not-Recently Used*)
- FIFO (*First-In First-Out*)
- Segunda chance
- Relógio (*Clock*)
- LRU (*Least-Recently Used*) / NFU (*Not-Frequently Used*)
- *Working Set*
- *WSClock*



Substituição de páginas

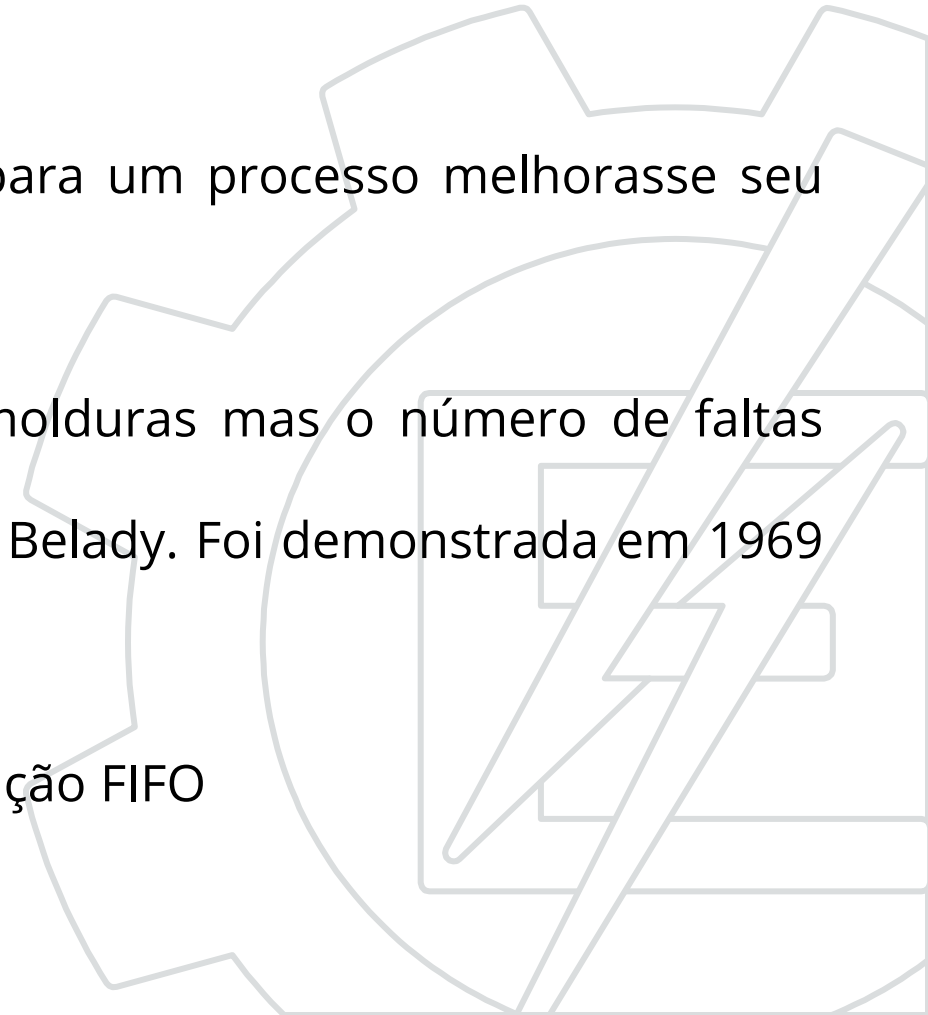
Anomalia de *Belady*



Gerenciamento de Memória

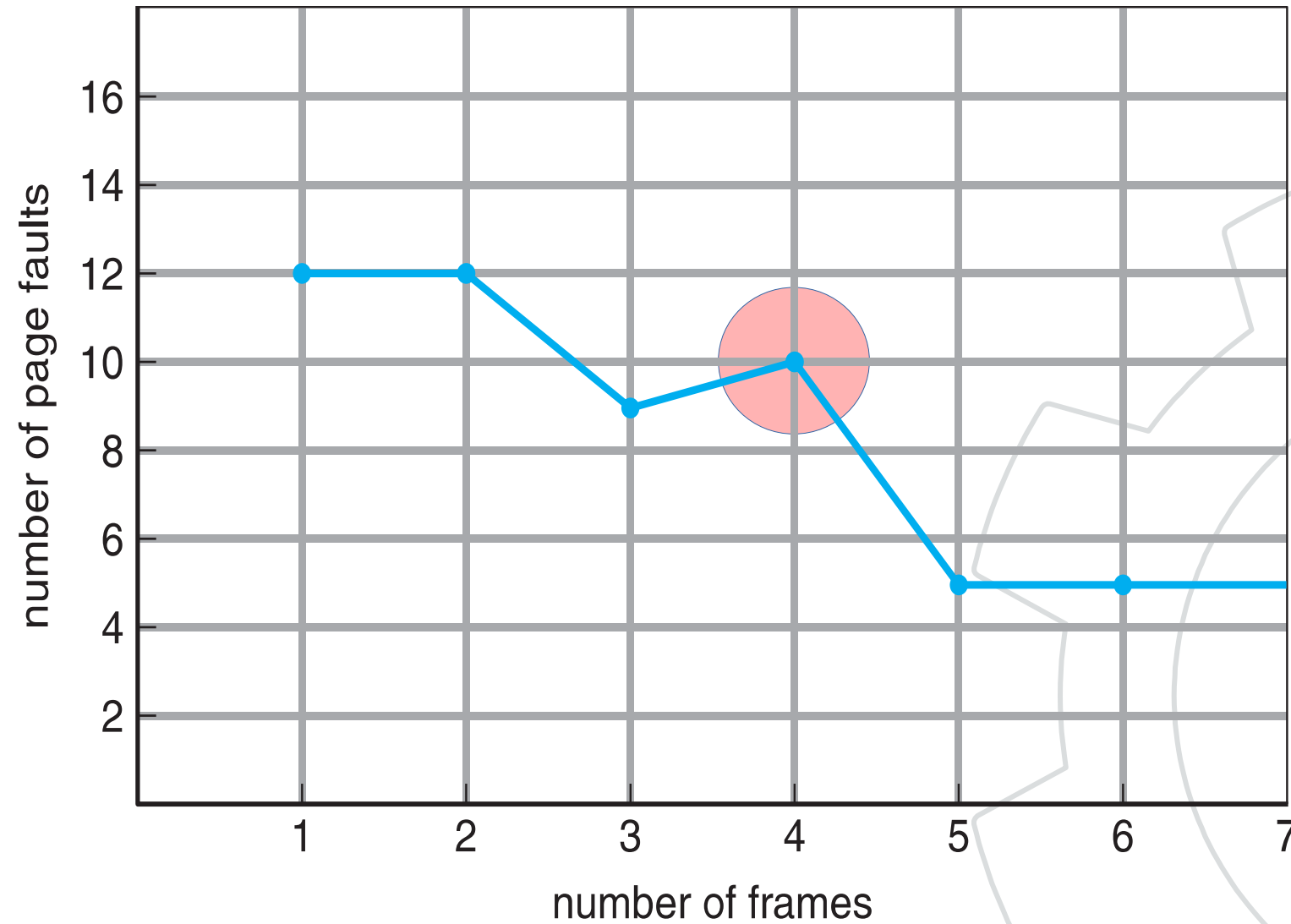
Substituição de páginas – Anomalia de *Belady*

- Em alguns algoritmos de substituição de páginas, a taxa de erros de página pode aumentar conforme o número de quadros alocados aumenta.
- O esperado seria que o fornecimento de mais memória para um processo melhorasse seu desempenho.
- Belady mostrou que é possível aumentar o número de molduras mas o número de faltas aumentar. A esse fenômeno dá-se o nome de Anomalia de Belady. Foi demonstrada em 1969 por László Belady.
- A anomalia é comum quando usado o algoritmo de substituição FIFO



Gerenciamento de Memória

Substituição de páginas – Anomalia de *Belady*



Gerenciamento de Memória

Substituição de páginas – Anomalia de *Belady*

All pages frames initially empty

		0	1	2	3	0	1	4	0	1	2	3	4
Youngest page		0	1	2	3	0	1	4	4	4	2	3	3
			0	1	2	3	0	1	1	1	4	2	2
Oldest page				0	1	2	3	0	0	0	1	4	4
		P	P	P	P	P	P				P	P	

9 Page faults

(a)

		0	1	2	3	0	1	4	0	1	2	3	4
Youngest page		0	1	2	3	3	3	4	0	1	2	3	4
			0	1	2	2	2	3	4	0	1	2	3
Oldest page				0	1	1	1	2	3	4	0	1	2
					0	0	0	1	2	3	4	0	1
		P	P	P	P			P	P	P	P	P	P

10 Page faults

(b)

(a) FIFO com 3 page frames. (b) FIFO com 4 page frames. Os P's representam page faults.

Bibliografia

- TANENBAUM, Andrew S; BOS, Herbert. Sistemas operacionais modernos. 4a ed. São Paulo: Pearson Education do Brasil, 2016.

Capítulo 3.

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/1233>

- DEITEL, H.M; DEITEL, P.J; CHOFFNES,D.R. Sistemas Operacionais. 3a ed. São Paulo: Pearson Prentice Hall, 2005. **Capítulo 10-11.**

<https://plataforma.bvirtual.com.br/Acervo/Publicacao/315>



Sistemas Operacionais

Prof. Otávio Gomes

otavio.gomes@unifei.edu.br

