

# SISTEMAS OPERACIONAIS

## AULA 08: MEMÓRIA VIRTUAL

2 de julho de 2025

Prof. Me. José Paulo Lima

IFPE Garanhuns



**INSTITUTO  
FEDERAL**  
Pernambuco

# SUMÁRIO



## Memória Virtual

Paginação

*Page fault*

## Algoritmos de Substituição de Página

Substituição aleatória

Ótimo

Não usada recentemente

FIFO

Segunda chance

Relógio

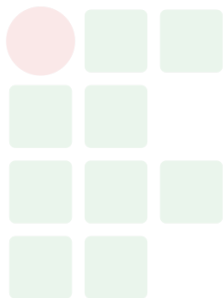
Menos Recentemente Usada

## Segmentação

Implementação

## Referências

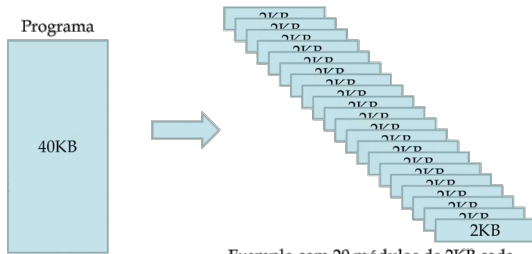
**MEMÓRIA VIRTUAL**



**INSTITUTO  
FEDERAL**  
Pernambuco

# MEMÓRIA VIRTUAL

- ▶ Início na década de 1960: sobreposições (*overlays*):
  - ▶ Programa dividido em módulos pelo programador;
  - ▶ Gerenciamento dos módulos pelo SO;
  - ▶ Divisão em módulos era enfadonho e difícil;
    - ▶ Gerava muitos *bugs*.
  - ▶ Automatização do processo.



Exemplo com 20 módulos de 2KB cada

# MEMÓRIA VIRTUAL



- ▶ O SO era responsável apenas pela troca de módulos na memória, seguindo as instruções executadas em cada módulo;
  - ▶ Definidas pelo programador;
  - ▶ Responsabilidade do programador.
- ▶ A divisão do programa em módulos era um trabalho lento e totalmente dependente da arquitetura;
  - ▶ Exemplo:
    - ▶ Se o programa fosse executado em uma máquina com menos memória se comparado a máquina que o mesmo foi projetado, ele deveria ser totalmente reorganizado.

# MEMÓRIA VIRTUAL



- ▶ Não demorou muito para a comunidade pensar em atribuir tal responsabilidade para o SO;
- ▶ Em 1961 surge o conceito de memória virtual:
  - ▶ Um programa maior que a memória primária pode ser executado por meio de uma escolha inteligente sobre qual parte vai estar na memória em cada instante;
  - ▶ Partes do programa podem ser dinamicamente carregadas para a memória primária.

# MEMÓRIA VIRTUAL



- ▶ A memória virtual tinha a ideia básica que cada programa tem seu próprio espaço de endereçamento que é dividido em blocos chamados páginas;
- ▶ As páginas são mapeadas na memória física mas nem toda página precisa estar na memória física;
- ▶ O hardware executa o mapeamento necessário dinamicamente;
- ▶ Quando um programa referencia uma parte de seu espaço de endereçamento que não está em sua memória física, o sistema operacional é alertado para obter a parte que falta e reexecutar a instrução que falhou.

# MEMÓRIA VIRTUAL

## PAGINAÇÃO



- ▶ Espaço de endereçamento do programa é dividido em páginas;
- ▶ Página:
  - ▶ Série contígua de endereços;
  - ▶ Mapeada na memória física;
  - ▶ Nem todas as páginas de um programa precisam estar em memória para a execução;
  - ▶ Se um endereço do espaço de um programa é referenciado e ele não está na memória:
    - ▶ SO deve obter a página correta e reexecutar a instrução que havia falhado.
- ▶ Os endereços gerados pelos programas são denominados endereços virtuais e constituem o espaço de endereçamento virtual;
- ▶ Em computadores que não tem memória virtual o endereço virtual é igual ao físico.



# MEMÓRIA VIRTUAL

## PAGINAÇÃO



- ▶ Em computadores com memória virtual é necessário um sistema especial para efetuar as conversões de endereços necessárias;
- ▶ Este módulo é denominado MMU (*Memory management Unit*), unidade de gerenciamento de memória, que mapeia endereços virtuais em endereços físicos.

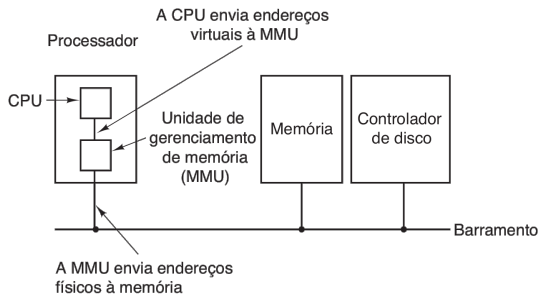


Figura extraída de Tanenbaum e Bos (2016, p. 135).

# MEMÓRIA VIRTUAL

## PAGINAÇÃO



- ▶ O espaço de endereçamento virtual é dividido em unidades denominadas páginas (*pages*);
- ▶ As unidades correspondentes na memória física são denominadas molduras de páginas (*frame pages*);
- ▶ As páginas e as molduras de páginas são sempre do mesmo tamanho;
- ▶ Exemplo:
  - ▶ 64KB de espaço de endereçamento virtual e 32 KB de memória física;
  - ▶ Podemos ter 16 páginas virtuais e 8 molduras de páginas.

# MEMÓRIA VIRTUAL

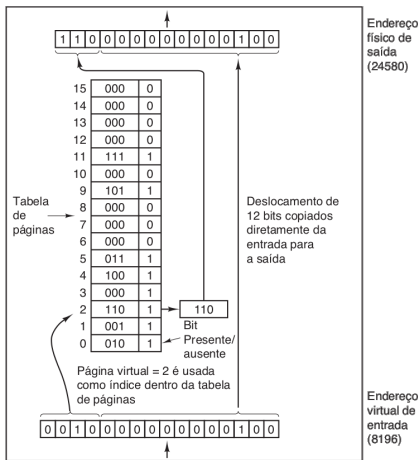
## PAGINAÇÃO: FUNCIONAMENTO



- ▶ Endereço virtual é enviado à MMU que detecta o mapeamento correspondente à moldura de página;
- ▶ A MMU transforma o endereço virtual em endereço físico e envia a memória;
- ▶ Um bit na tabela de páginas indica se a página está fisicamente presente na memória.

# MEMÓRIA VIRTUAL

## PAGINAÇÃO: FUNCIONAMENTO



- ▶ Quando o endereço virtual é apresentado à MMU para tradução o hardware primeiro verifica se o número da sua página virtual está presente na TLB (*Translation lookside buffer*);
- ▶ Compara todos as entradas da TLB simultaneamente em paralelo.

# MEMÓRIA VIRTUAL

## PAGINAÇÃO: FUNCIONAMENTO

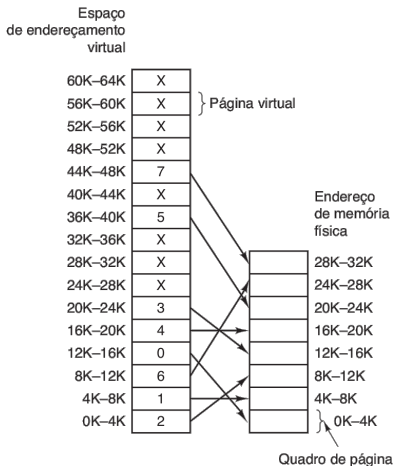


Figura extraída de Tanenbaum e Bos (2016, p. 136).

- ▶ A instrução `mov REG, 0` é convertida pela MMU para `mov REG, 8192`:
  - ▶ O endereço 0 (virtual) pertence a moldura 2 da memória física (real).
- ▶ A instrução `mov REG, 8192` é convertida pela MMU para `mov REG, 24576`:
  - ▶ O endereço 8192 pertence a moldura 6 da memória física.

# MEMÓRIA VIRTUAL

## PAGINAÇÃO: FUNCIONAMENTO

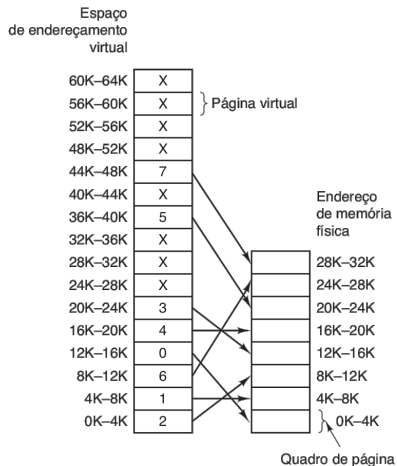


Figura extraída de Tanenbaum e Bos (2016, p. 136).

► A instrução `mov REG, 32780` a MMU encontra o indicador “X”:

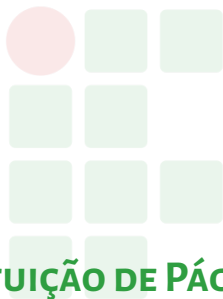
- Isso significa que o conteúdo não está na memória principal, está no disco;
- Isso é conhecido como *page fault* (falta de página).

# MEMÓRIA VIRTUAL

PAGINAÇÃO: *PAGE FAULT*



- ▶ A MMU deve trazer a página do disco para a memória primária;
- ▶ Caso todas as molduras estão sendo referenciadas pela memória virtual, a MMU deve executar um algoritmo para substituição de página;
  - ▶ Elimina (armazena no disco) uma das páginas da memória física (moldura) e traz a página requisitada para a posição liberada;
  - ▶ Alguns algoritmos existentes:
    - ▶ Não usada recentemente;
    - ▶ FIFO;
    - ▶ Menos recentemente utilizada (MRU).
  - ▶ É importante notar que este problema se repete em diferentes área da computação:
    - ▶ *Cache* de servidores web;
      - Quais páginas devem estar no cache do servidor Apache?
    - ▶ *Cache* do *proxy*;
    - ▶ *Cache* do YouTube.



**ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA**

**INSTITUTO  
FEDERAL**  
Pernambuco



# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## SUBSTITUIÇÃO ALEATÓRIA



### Definição:

“Todas as páginas alocadas na memória principal têm a mesma chance de serem selecionadas, inclusive os frames que são frequentemente referenciados.” (MACHADO; MAIA, 2017, p. 186).

- ▶ Escolha aleatória de uma moldura de página na memória principal para ser substituída;
- ▶ Fácil implementação;
- ▶ Desempenho frustrante.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## ÓTIMO



- ▶ Remove a página que será referenciada mais tardiamente;
  - ▶ Ou seja, evita ao máximo futuras faltas de página.
- ▶ Existe, mas é impossível de implementar em um Sistema Operacional de uso geral.
  - ▶ Então é como se não existisse.

### Explicação:

“[...] é impossível ser implementado, pois o sistema operacional não tem como conhecer o comportamento futuro das aplicações. Essa estratégia é utilizada apenas como modelo comparativo na análise de outros algoritmos de substituição.” (MACHADO; MAIA, 2017, p. 186).

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## ÓTIMO

Espaço  
de endereçamento  
virtual

|         |   |
|---------|---|
| 60K–64K | X |
| 56K–60K | X |
| 52K–56K | X |
| 48K–52K | X |
| 44K–48K | 7 |
| 40K–44K | X |
| 36K–40K | 5 |
| 32K–36K | X |
| 28K–32K | X |
| 24K–28K | X |
| 20K–24K | 3 |
| 16K–20K | 4 |
| 12K–16K | 0 |
| 8K–12K  | 6 |
| 4K–8K   | 1 |
| 0K–4K   | 2 |

} Página virtual

Endereço  
de memória  
física

|  |         |     |
|--|---------|-----|
|  | 28K–32K | 300 |
|  | 24K–28K | 200 |
|  | 20K–24K | 100 |
|  | 16K–20K | 50  |
|  | 12K–16K | 10  |
|  | 8K–12K  | 500 |
|  | 4K–8K   | 350 |
|  | 0K–4K   | 230 |

Quadro de página

} Instante que as páginas  
serão referenciadas

## Algoritmo Ótimo:

Ao escolher a moldura de página 2, o sistema evitará ao máximo a próxima falta de página.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## NÃO USADA RECENTEMENTE

- ▶ A tabela de páginas precisa de dois bits para cada registro de página:
  - ▶ R - Indica se a página foi referenciada;
  - ▶ M - Indica se a página foi modificada.

| Categorias | Bits avaliados  | Resultado                                |
|------------|-----------------|--|
| 1          | BR = 0 e BM = 0 | Página não referenciada e não modificada |
| 2          | BR = 0 e BM = 1 | Página não referenciada e modificada     |
| 3          | BR = 1 e BM = 0 | Página referenciada e não modificada     |
| 4          | BR = 1 e BM = 1 | Página referenciada e modificada         |

Tabela extraída de Machado e Maia (2017, p. 188).

- ▶ O algoritmo NUR remove aleatoriamente uma página da classe de mais baixa ordem que não esteja vazia;
- ▶ As classes embutem informações sobre as páginas mais “importantes” (mais utilizadas) e faz uma escolha “inteligente” sobre a eliminação de uma página.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## NÃO USADA RECENTEMENTE

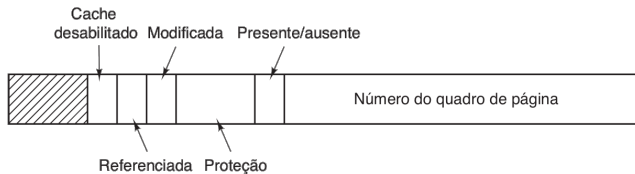


Figura extraída de Tanenbaum e Bos (2016, p. 138).

- ▶ Tabela de páginas:
  - ▶ Estrutura de uma entrada de tabelas de páginas.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## FIFO



- ▶ A primeira página da fila é a próxima a deixar a memória principal na próxima falta de página;
  - ▶ Fácil implementação;
  - ▶ Baixo custo computacional;
  - ▶ Não tem garantia alguma de qualidade;
  - ▶ Sua implementação pura é raramente encontrada na implementação de caches.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## SEGUNDA CHANCE



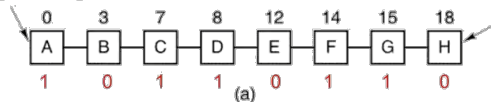
- ▶ É uma simples modificação do algoritmo FIFO;
  - ▶ Utiliza a fila, mas evita eliminar uma página recentemente utilizada.
- ▶ Para analisar esta informação, é utilizado o bit R (referenciada);
- ▶ Se a página foi lida ou referenciada recentemente, ela não é removida da lista;
- ▶ Vai para o fim da lista, e seu bit R (recentemente referenciada) passa a ser 0;
- ▶ Mescla a ideia do FIFO, mas inserindo informações de popularidade

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

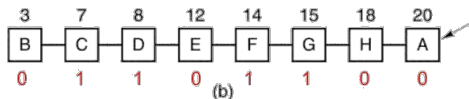
## SEGUNDA CHANCE



Primeira página carregada



Página mais recentemente carregada



Página A é tratada como página mais recentemente carregada

- (a) Lista de páginas em ordem FIFO;
- (b) Estado da lista em situação de falta de página no instante 20, com o bit R da página A em 1 (números representam instantes de carregamento das páginas na memória)



# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## RELÓGIO



- ▶ Melhoria do Algoritmo de Segunda Chance;
  - ▶ Ao invés de implementar uma fila, implementa uma lista circular:
    - ▶ Evita remover elementos do início da fila, e inserir no fim da fila;
    - Evita alocação de memória dinâmica.
    - ▶ Apenas move seu apontador do primeiro da “fila”.
  - ▶ Só é eficiente porque a memória virtual tem um tamanho fixo.

# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## RELÓGIO

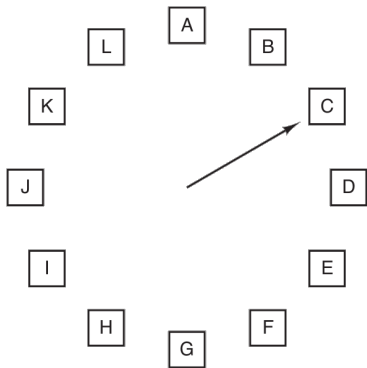


Figura extraída de Tanenbaum e Bos (2016, p. 147).

- ▶ Quando ocorre uma falta de página, a página indicada pelo ponteiro é inspecionada. A ação executada depende do bit R:
  - ▶  $R = 0$ :
    - ▶ Remover a página.
  - ▶  $R = 1$ :
    - ▶ Zerar R e avançar o ponteiro.

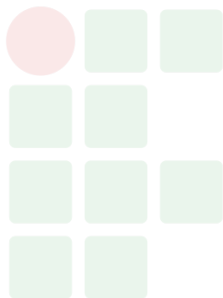
# ALGORITMOS DE SUBSTITUIÇÃO DE PÁGINA

## MENOS RECENTEMENTE USADA



- ▶ Utiliza da hipótese:
  - ▶ Páginas referenciadas intensamente na últimas instruções provavelmente serão de novo referenciadas de maneira intensa nas próximas instruções;
  - ▶ Obviamente, remove a página a mais tempo não referenciada;
    - ▶ Observe que isso não acontece no algoritmo de Segunda Chance.
- ▶ Vantagem:
  - ▶ Na ocorrência de uma falta de página, a página a mais tempo sem utilização é a primeira da lista.
- ▶ Desvantagem:
  - ▶ A lista tem que ser gerenciada a cada referencia à memória.

**SEGMENTAÇÃO**



**INSTITUTO  
FEDERAL**  
Pernambuco

# MEMÓRIA VIRTUAL

## SEGMENTAÇÃO



25

A memória virtual até agora é unidimensional porque o endereço virtual vai de 0 a algum endereço máximo, um após o outro.

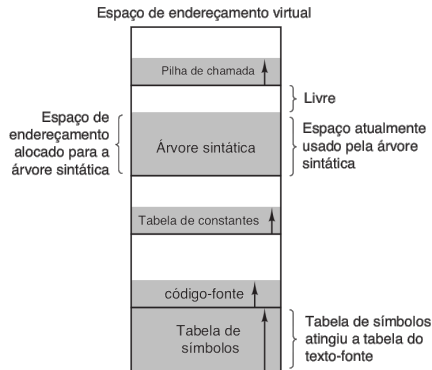


Figura extraída de Tanenbaum e Bos (2016, p. 166).

# MEMÓRIA VIRTUAL

## SEGMENTAÇÃO

Para muitos problemas, ter dois ou mais espaços de endereçamento separados pode ser muito melhor do que ter somente um.

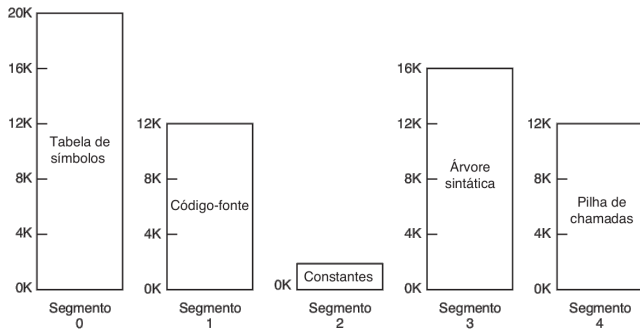


Figura extraída de Tanenbaum e Bos (2016, p. 167).

# MEMÓRIA VIRTUAL

## SEGMENTAÇÃO



- ▶ Imagine um programa grande com um número imenso de variáveis, a região do espaço de endereçamento pode se esgotar;
- ▶ Outra possibilidade é imitar o Robin Hood, tirar espaços das tabelas com excesso de entradas livres;
- ▶ É necessário uma maneira de livrar o programador a obrigatoriedade de gerenciar a expansão e a contração de tabelas;
- ▶ Uma solução extremamente abrangente e direta é prover a máquina com muitos espaços de endereçamento completamente independentes chamados “segmentos”;
- ▶ Cada segmento é constituído de uma sequência de limiar de endereços, de 0 a algum máximo;
- ▶ Segmentos diferentes podem ter diferentes tamanhos durante a execução.

# MEMÓRIA VIRTUAL

## SEGMENTAÇÃO: IMPLEMENTAÇÃO



- ▶ A implementação da segmentação difere da paginação em um ponto essencial: as páginas têm tamanhos fixos e os segmentos não;
- ▶ A memória está dividida em regiões, algumas com segmentos e outras com lacunas, chamadas fragmentação externa.



# MEMÓRIA VIRTUAL

## SEGMENTAÇÃO: IMPLEMENTAÇÃO

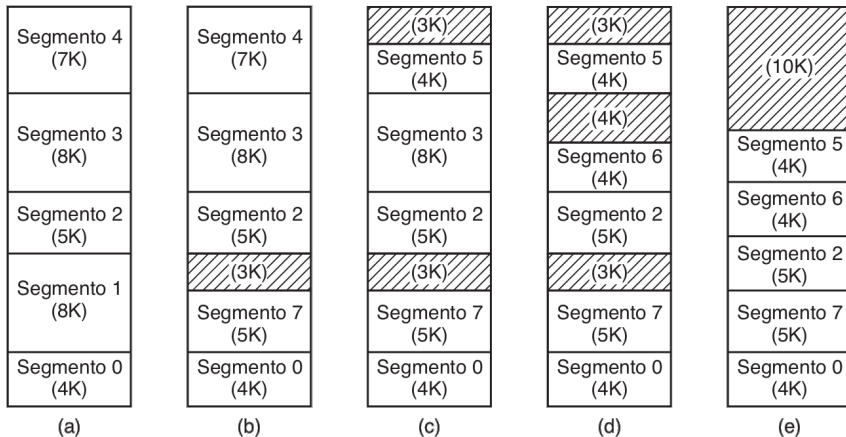
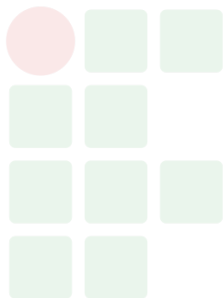


Figura extraída de Tanenbaum e Bos (2016, p. 169).



## REFERÊNCIAS

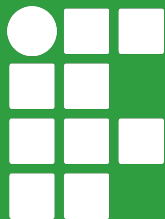


INSTITUTO  
FEDERAL  
Pernambuco

# REFERÊNCIAS I



-  MACHADO, Francis Berenger; MAIA, Luiz Paulo. **Arquitetura de Sistemas Operacionais**. 5. ed. Rio de Janeiro: LTC, 2017. ISBN 978-85-216-2210-9.
-  TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016.



**INSTITUTO FEDERAL**

Pernambuco