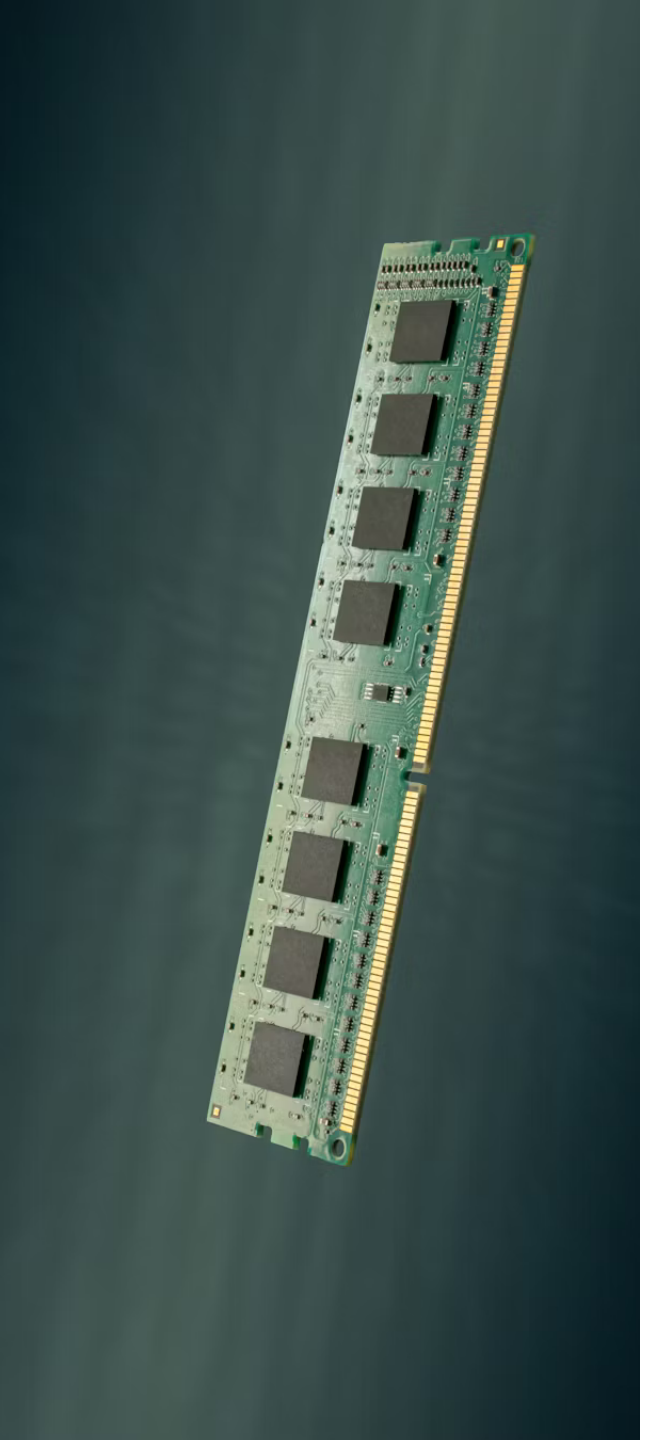


Gerenciamento de Memória

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizado

Ao final desta aula, você será capaz de:

- Compreender os mecanismos e usos de memória virtual
- Entender operações de tradução de endereços lógicos em físicos
- Aplicar algoritmos de substituição de páginas

Disclaimer

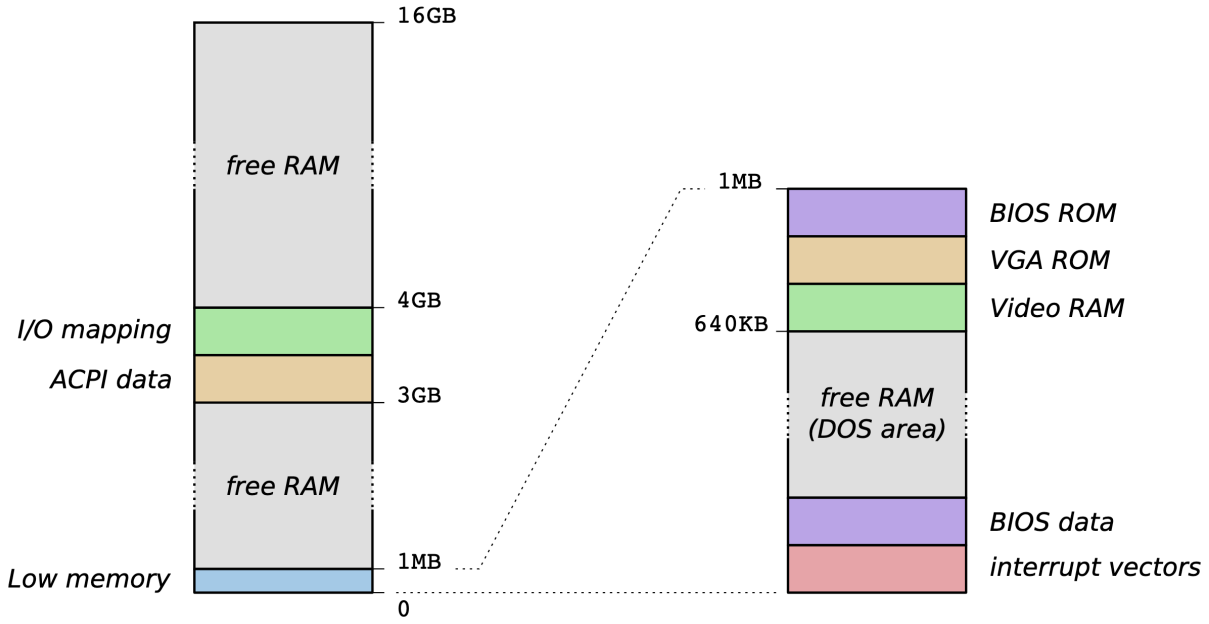
Parte do material apresentado a seguir foi adaptado de:

- IT Systems – Open Educational Resource, produzido por Jens~Lechtenböger da Universidade de Münster; e
- Sistemas Operacionais: Conceitos e Mecanismos produzido por Prof. Carlos A. Maziero, da UFPR.

Imagens decorativas retiradas de Unsplash

Memória principal

- **Espaço de memória física:** A quantidade de memória RAM disponível.

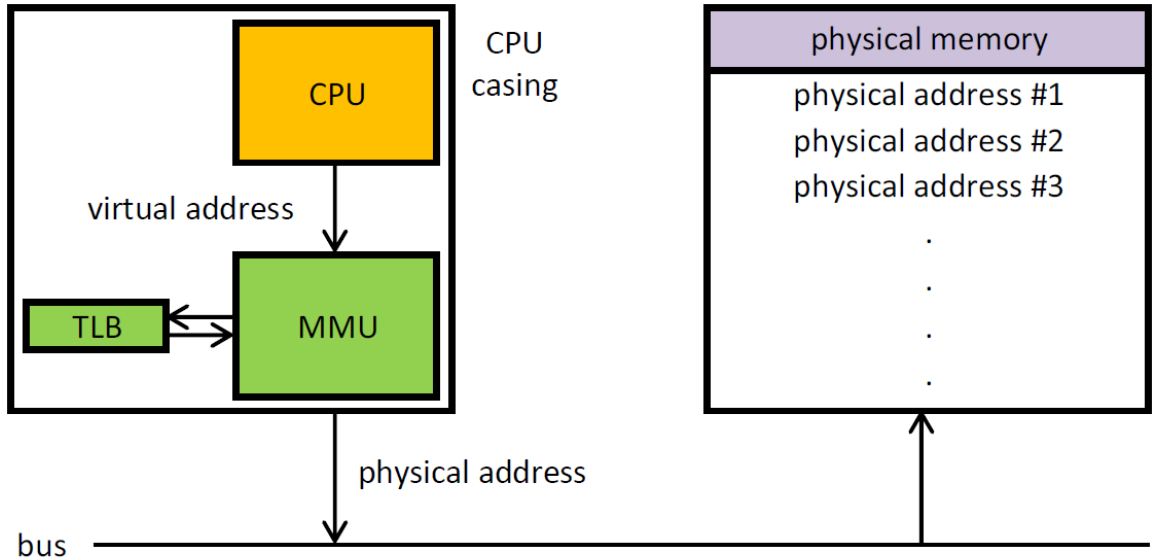


Layout da memória física de um computador. Fonte da Imagem: Prof. Maziero

- **Espaço de endereçamento:** O conjunto de endereços de memória que um processador pode produzir.
  - *Intel 80386* 32 bits, bus de 32 bits: 2^{32} endereços (4 GB)
  - *Intel Core i7* 64 bits, bus de 48 bits : 2^{48} endereços (256 TB)
- Espaço de endereçamento é **independente** da quantidade de memória RAM

Memória Virtual

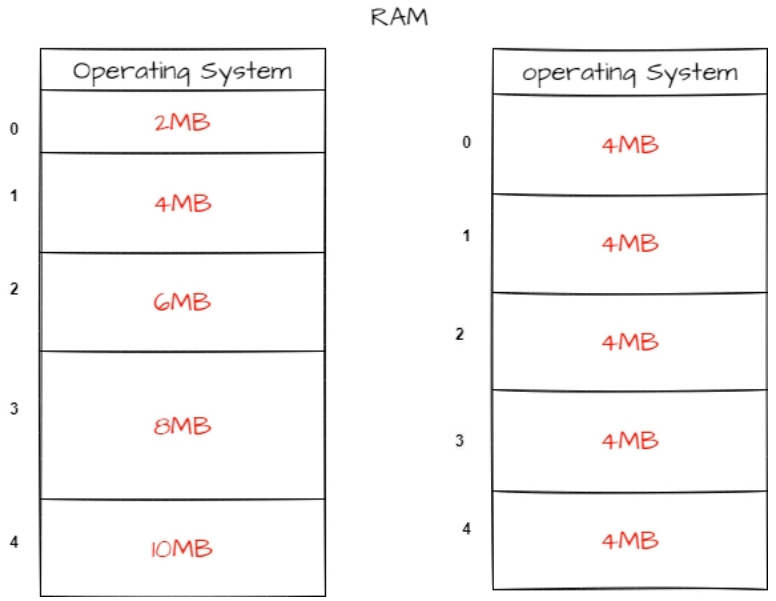
- Endereços **físicos**: endereços acessados na RAM.
- Endereços **lógicos**: endereços usados pela CPU.
- Ao executar, a CPU e os processos só veem endereços lógicos.
  - Simplifica o uso da memória pelo SO e processos.
  - CPU não acessa diretamente endereços de memória física $\longrightarrow$ Requisita da Memory Management Unit (MMU) endereços virtuais
  - MMU traduz endereços virtuais em endereços físicos (**extremamente rápida**)
- A RAM física pode ser mapeada em vários processos ao mesmo tempo
- Fonte da imagem: Wikipedia



CPU: Central Processing Unit
MMU: Memory Management Unit
TLB: Translation lookaside buffer

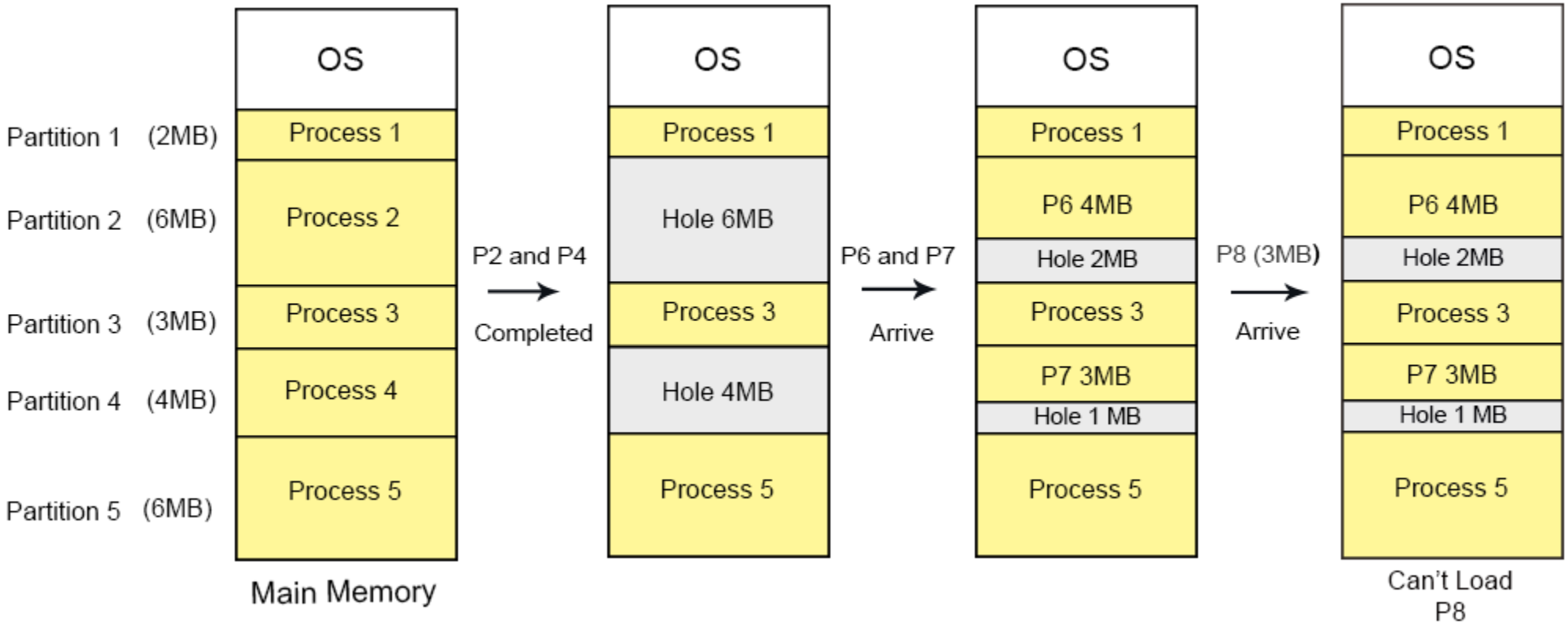
Memória Virtual por Partições

- Área de RAM **contígua** (sem buracos) com T bytes
  - Endereços lógicos no intervalo [0, T–1]
  - As partições podem ser tamanhos iguais ou distintos
- Cada partição é ocupada por um processo
- SO mantém tabela indicando quais partes da memória estão disponíveis e quais estão ocupada



Esquerda: Partições com diferentes tamamnhos. Direita: Partições com tamanhos iguais.
Fonte da Imagem: Fixed partition @Code360

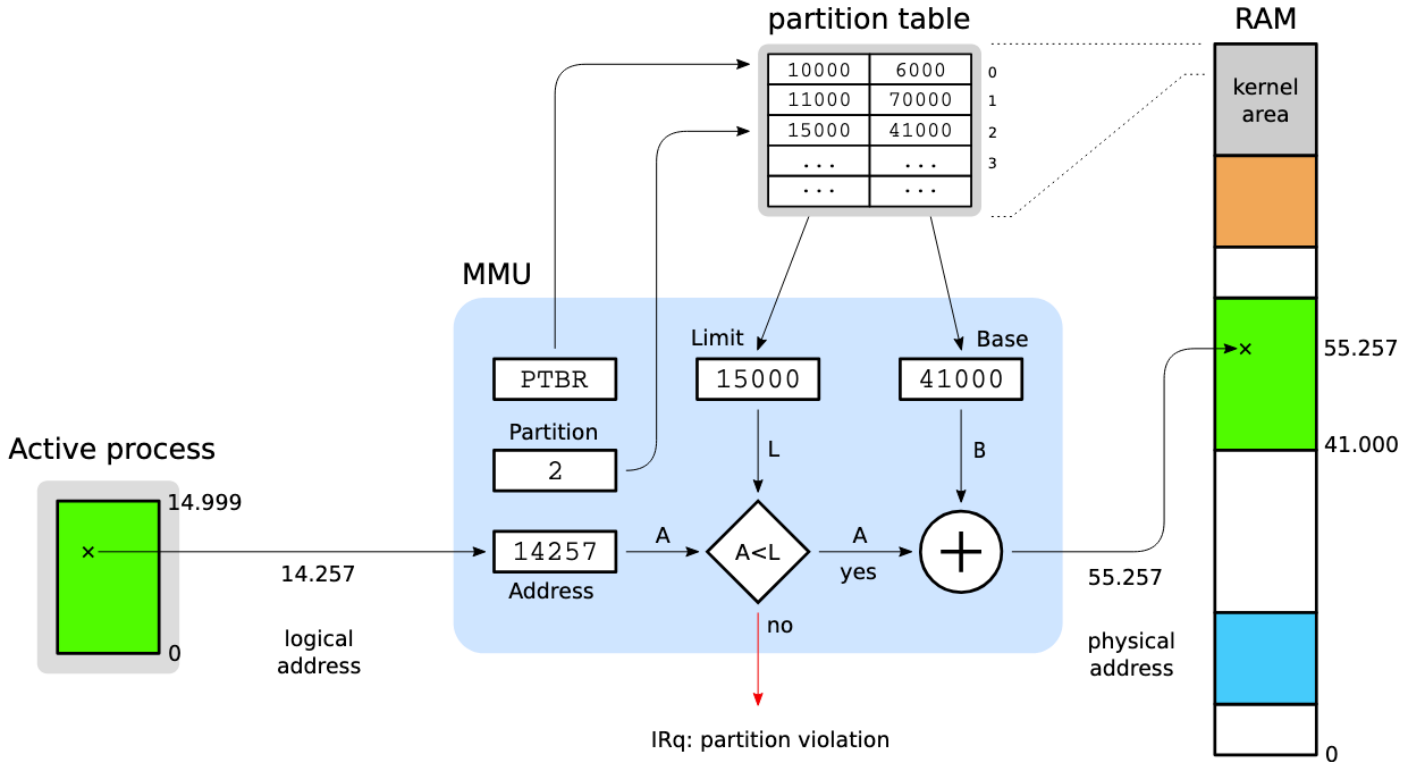
Memória Virtual por Partições (cont.)



Fonte da Imagem: Dynamic partition @CSTaleem

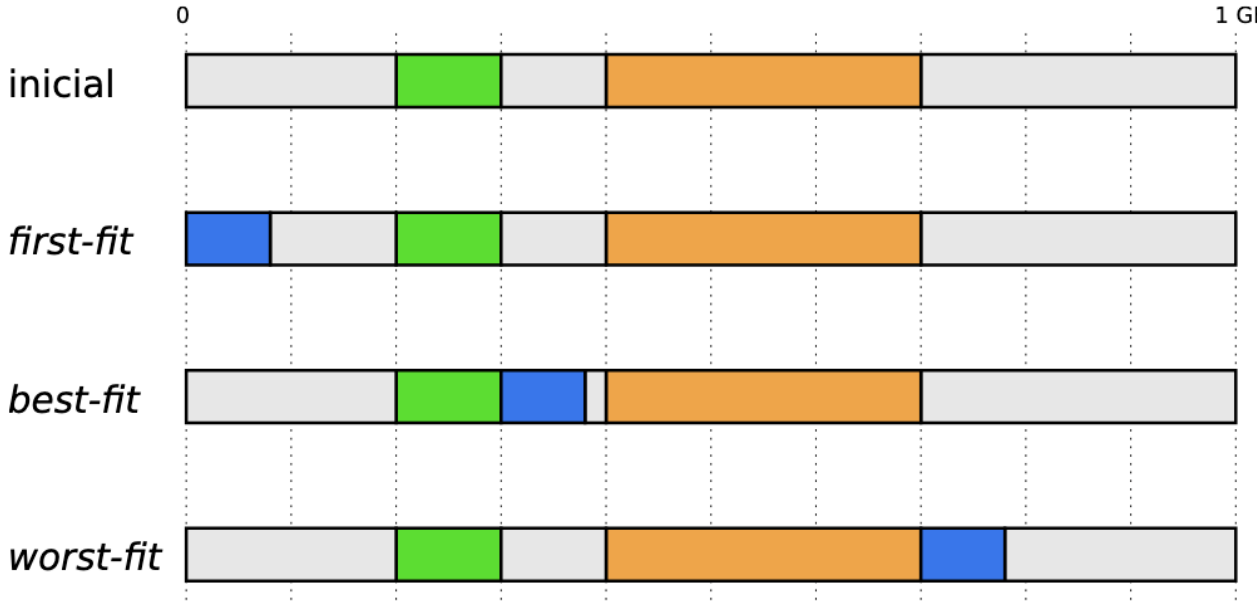
Tradução de Endereços pela MMU

- *Registrador Base* (B): Endereço físico inicial da partição
- *Registrador Limite* (L): Tamanho em bytes da partição
- Proteção: Se o endereço lógico $\geq$ L, então a MMU gera uma Interrupt Request (IRq)
- Os valores de B e L são ajustados a cada troca de contexto
- Na Imagem: Funcionamento da MMU com partições. Fonte: Prof. Maziero



Estratégias de Alocação de Memória

- **Framentação**: *buracos* na memória
- *Objetivo*: Minimizar a ocorrência de fragmentação externa
- *First-fit*: Escolher a primeira área livre que satisfaça o pedido de alocação
- *Best-fit*: Escolher a menor área possível que possa receber a alocação, minimizando o desperdício de memória
- *Worst-fit*: Escolher sempre a maior área livre possível
- Mais detalhes no material adicional
- Fonte da imagem: Prof. Maziero



Organização de memória em Páginas

- Espaço de endereçamento divido em blocos
  - Geralmente, 4096 bytes (4 KB)
- Memória física dividida em blocos (*frames*) de mesmo tamanho
- Mapeamento: em qual frame está cada página
  - Uma página pode estar em qualquer posição da memória física
  - Um processo consiste em múltiplas páginas
  - Cada processo com sua tabela de páginas
  - *Page Table Base Register* (PTBR): registrador da MMU que referencia a tabela de páginas do processo em execução
- Fonte da Imagem: OS @KU Leuven

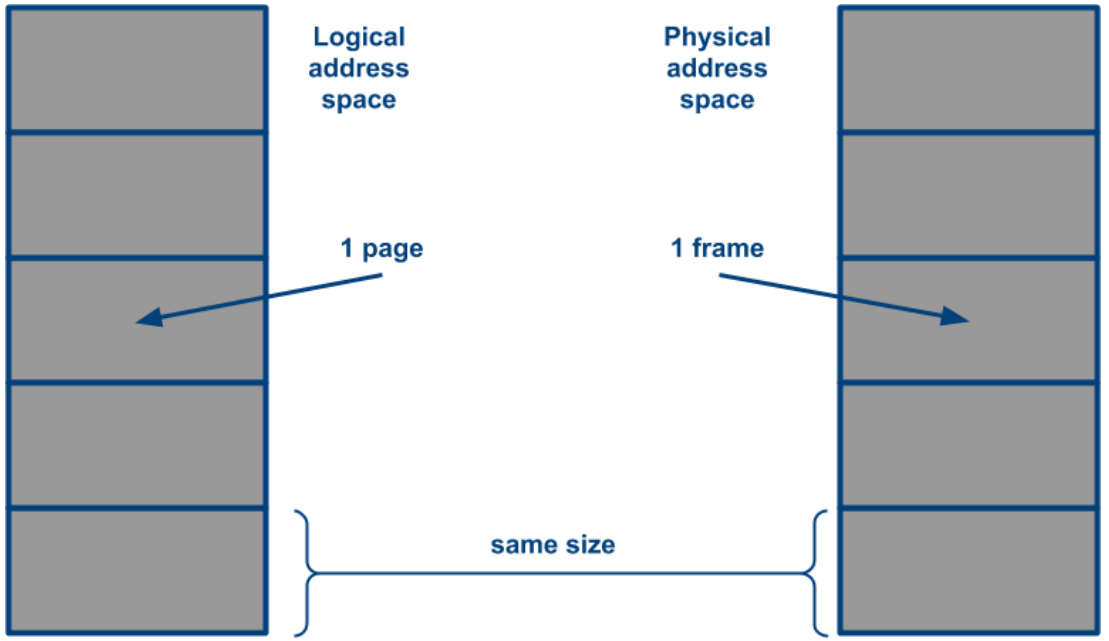
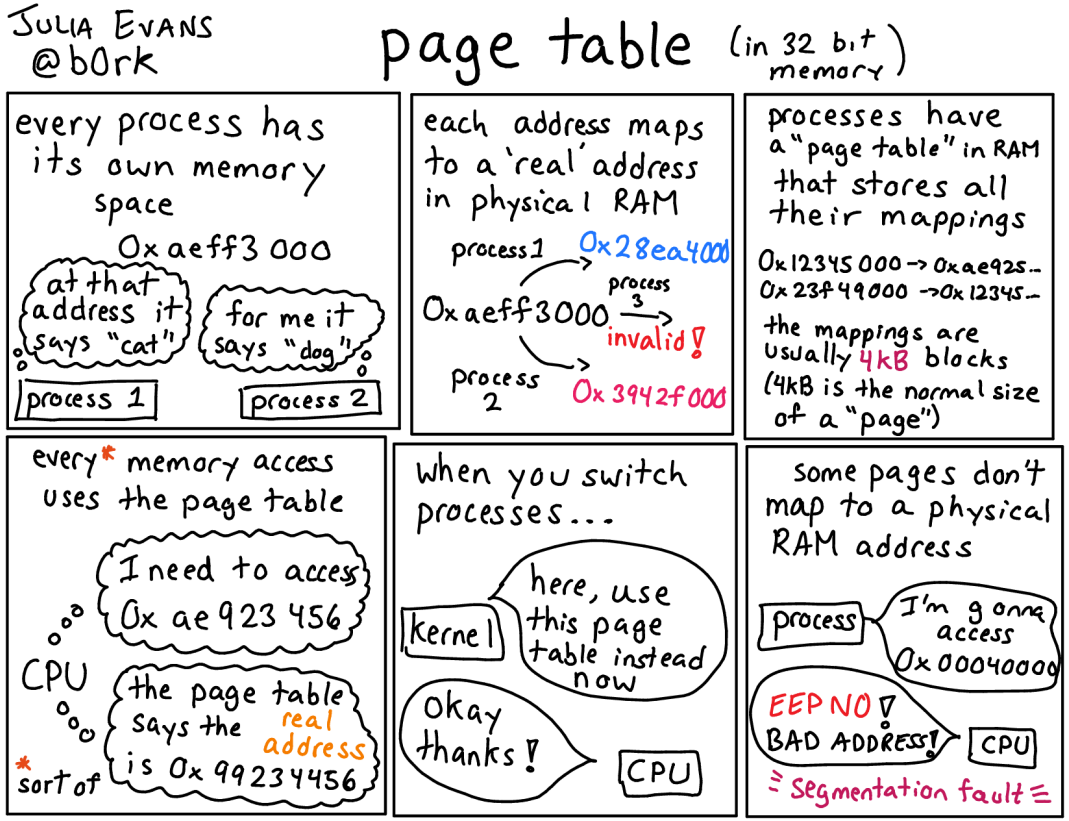


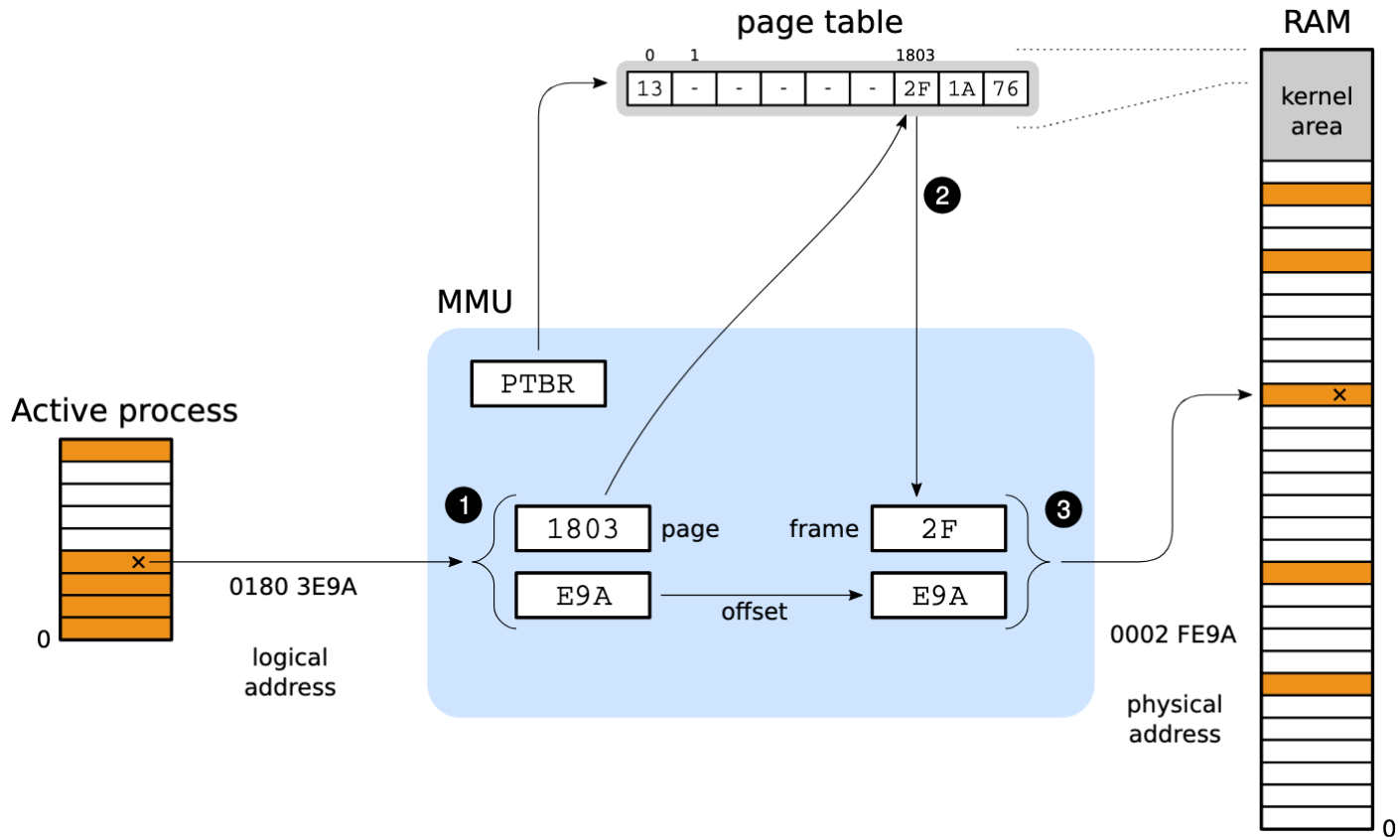
Tabela de Páginas

- Uma página sempre tem 2^m bytes de tamanho
  - n bits menos significativos definem o offset (posição do endereço dentro da página)
  - $m - n$ bits restantes definem o número da página
- Exemplo: CPU de 32 bits, páginas de 4 KB, temos:
  - $\mathbf{0000\ 0001\ 1000\ 0000\ 0011}\ 1110\ 1001\ 1010$
  - $4\ KB = 2^{12} \rightarrow$ *12 bits de offset*
  - $32 - 12 = \mathbf{20}$ **bits de número de página**
  - Ou seja, 2^{20} páginas
- Fonte da Imagem: JVNS



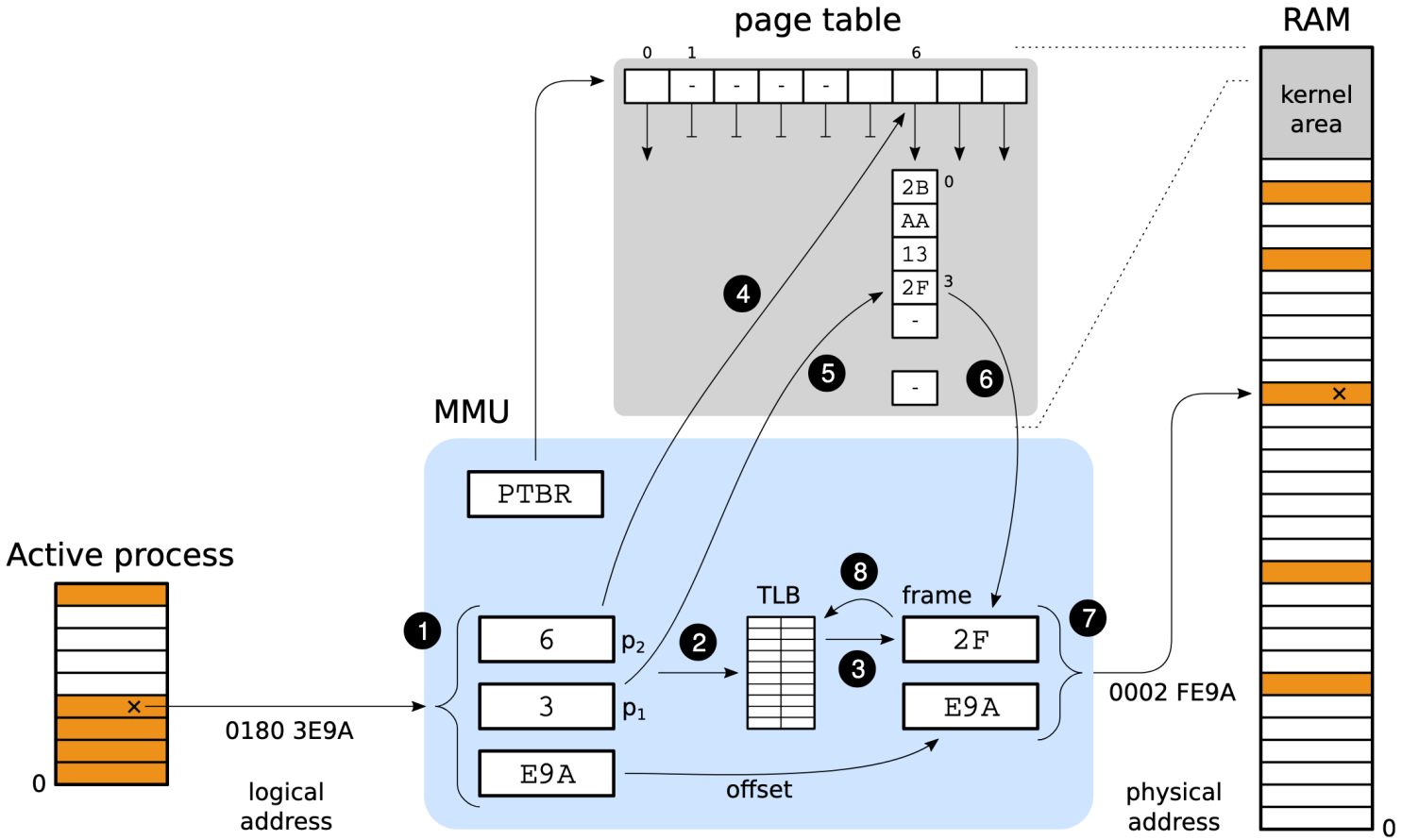
MMU com Paginação

- **1**. Analisar o endereço lógico: página e offset
- **2**. Consultar a tabela de páginas
- **3**. Formar o endereço físico: frame e offset
- Fonte da Imagem: Prof. Maziero



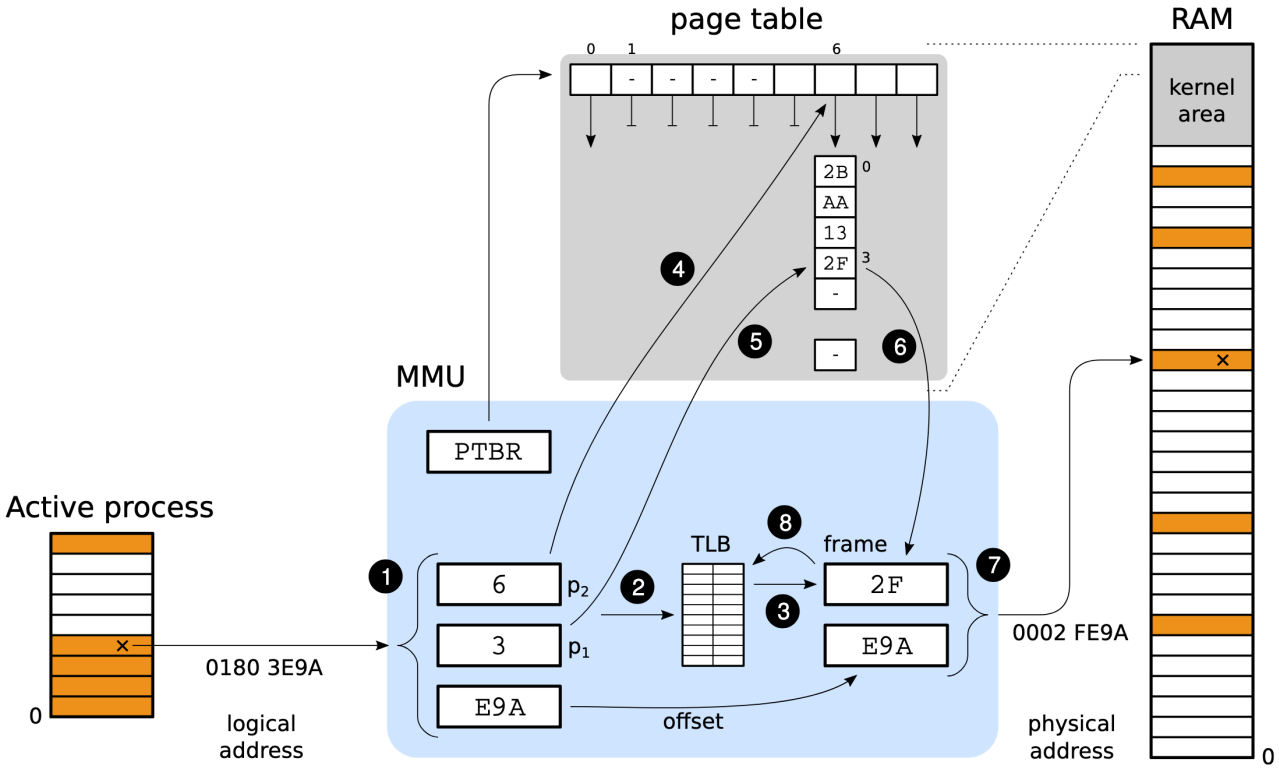
Cache de Tabela de Páginas

- Tabelas de Páginas armazenadas na memória RAM
- Problema: Tempo de acesso é longo (comparado com o tempo da CPU)
- Solução: Manter a tabela em memória cache dentro da MMU
- *Translation Lookaside Buffer* (TLB): armazena pares {página, frame} obtidos em consultas recentes
- Fonte da Imagem: Prof. Maziero



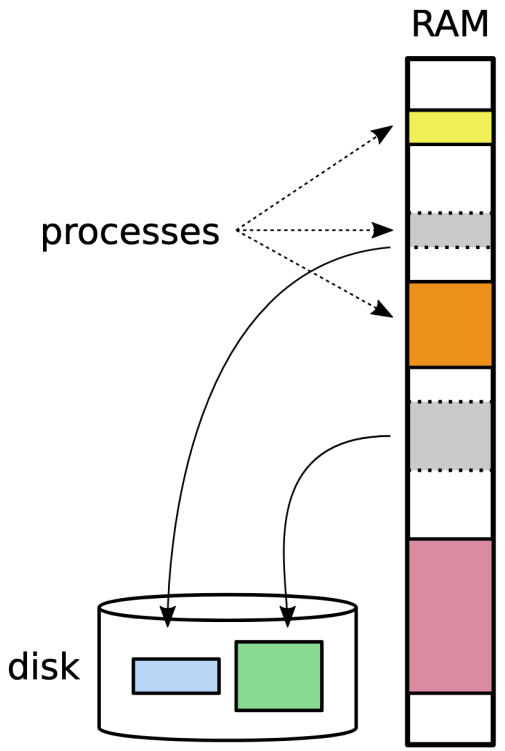
Cache de Tabela de Páginas

- **1**. Analisa o endereço lógico: página e offset
- **2**. Verifica se {página, frame}
  - **3**. Sim: *TLB hit*
  - **4**. Não: *TLB miss*, busca na tabela de páginas (passos **4** a **6**)
- **7**. Forma o endereço físico: frame e offset
- **8**. Adiciona {página, frame} ao TLB
- Impacto na eficiência e **Localidade de referências**: Propriedade de um processo ou sistema concentrar seus acessos em poucas áreas da memória
  - Localidade temporal
  - Localidade espacial
  - Localidade sequencial
- Fonte da Imagem: Prof. Maziero

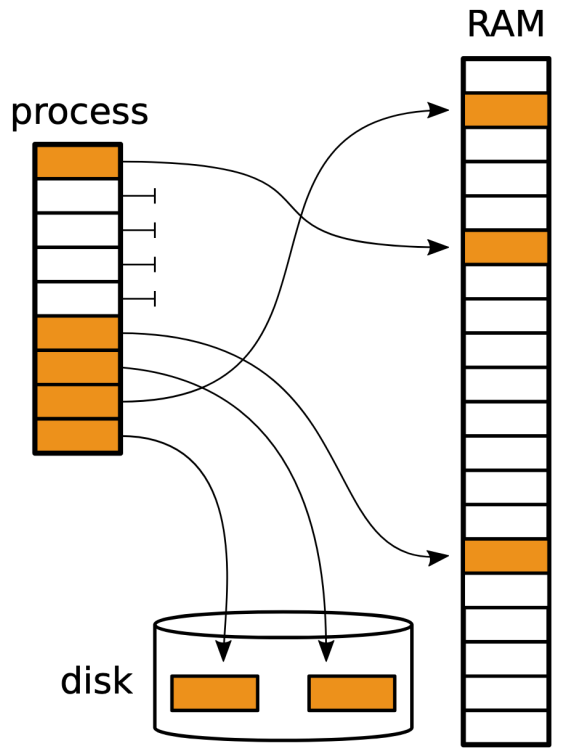


Paginação em Disco

- Acessar Memória Secundária como uma extensão da memória RAM
- **Swapping**: Mover um processo ocioso da RAM para o disco (swap-out), liberando memória para outros processos. Quando o processo volta a ficar ativo, ele é carregado de volta para a RAM (swap-in).
- **Paging** (paginação): Mover páginas para o disco (p*age-out*). Se o processo tentar acessar uma dessas páginas, o SO carrega a página faltante volta (*page-in*).
  - Paging mais eficiente que Swapping
  - Mais usada em SOs modernos
- Fonte da Imagem: Prof. Maziero



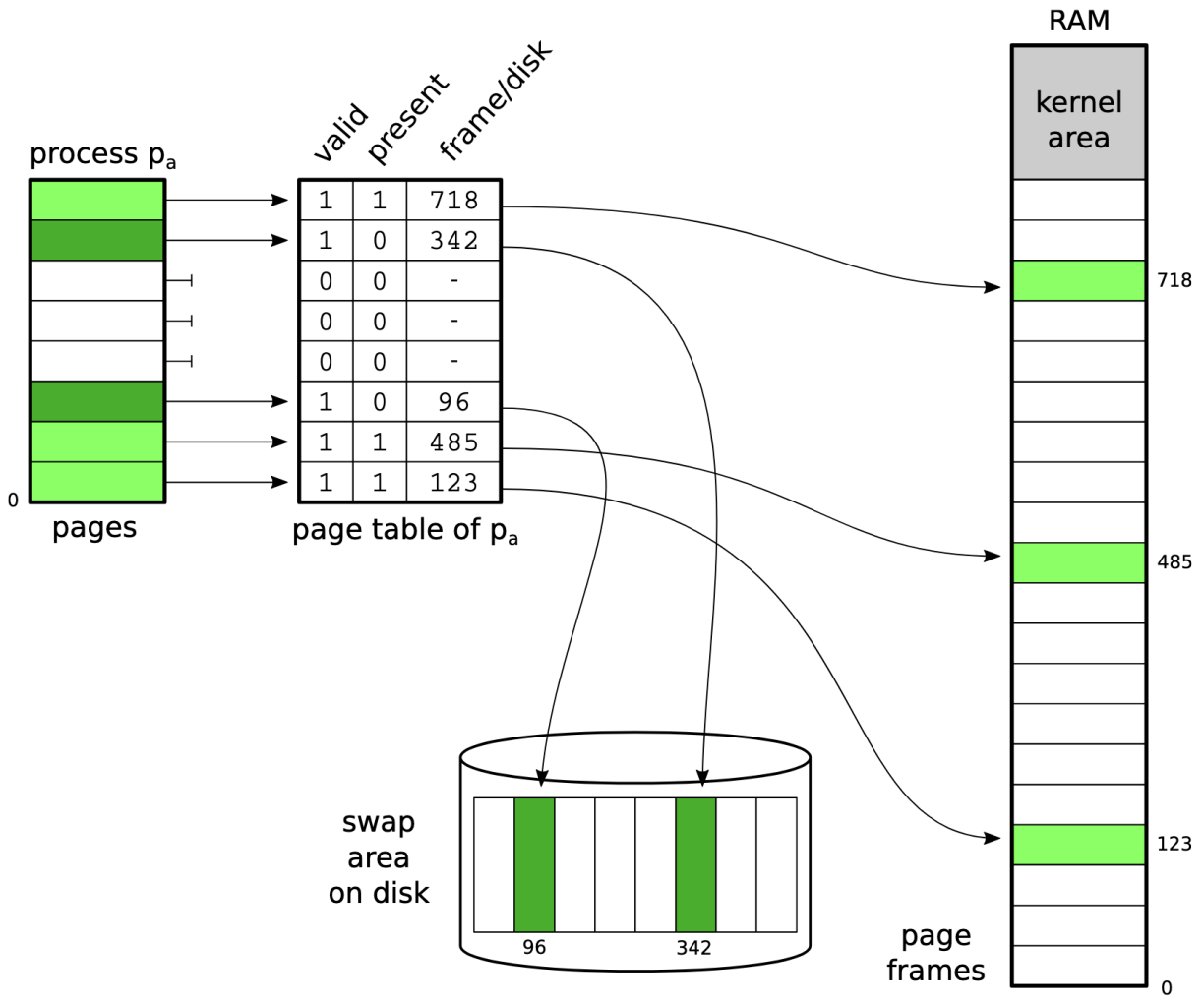
swapping



paging

Paginação em Disco (cont.)

- Kernel decide que páginas manter em RAM
  - Move páginas RAM <—> Disco
- Se página faltante for acessada, MMU gera interrupção *page fault*
  - Um endereço virtual válido é acessado, mas nenhum frame físico é alocado
  - Um endereço virtual inválido é acessado
  - Kernel move a página do disco de volta para RAM
- **Problema**: Disco é lento. Quais páginas manter para não perder desempenho?
  - Cerca de 20 milhões de acessos à memória por segundo
- Fonte da Imagem: Prof. Maziero



Tratadores de Falha de Página

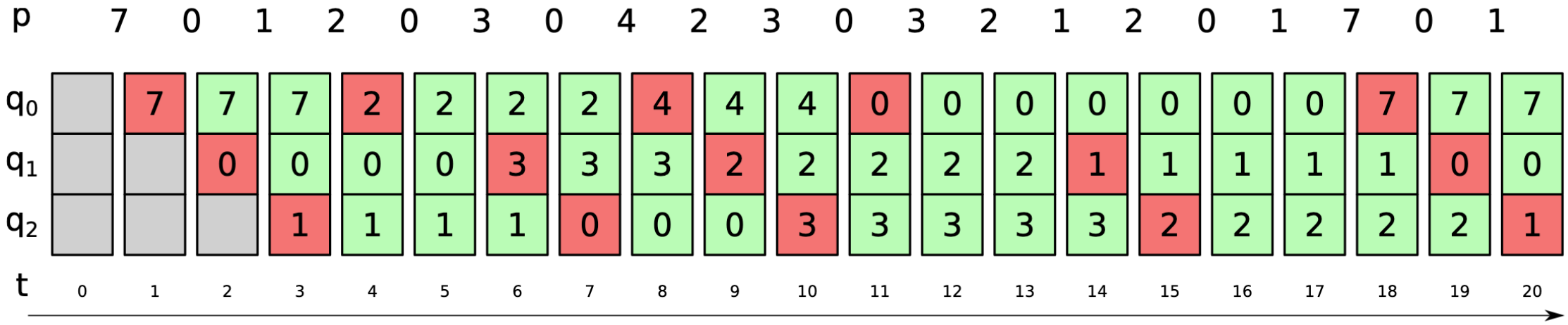
- Se o endereço virtual faz parte da tabela de páginas, ou seja, existe um frame físico correspondente para ele, o MMU encaminha o endereço físico para o subsistema de memória
- Se não, uma falha de página é gerada
- As falhas de página são tratadas por rotinas pré-registradas do sistema operacional: **tratadores de falha de página**
- Tratador (*handler*) de falha de página
  - a falha de página causa a execução do tratador de falha de página (parte do sistema operacional)
  - O tratador verifica se o endereço virtual é válido, ou seja, foi alocado; emite uma exceção de memória (ou seja, *segmentation fault*) se não
  - se o endereço é válido, verifica se já existe um frame físico existente para a página virtual
- A tabela de páginas é atualizada com os novos mapeamentos
- A instrução que causou a falha de página é executada novamente

Algoritmos de Paginação

- Cadeias de referências: sequência de páginas acessadas por um processo ao longo de sua execução
- Considere a cadeia a seguir com 20 acessos a 6 páginas: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1
- Somente o primeiro acesso em cada grupo de acessos consecutivos à mesma página provoca uma falta de página
- Algoritmo ótimo (**OPT**): Solução teórica, não é implementável
    - Define o limite teórico para comparação de algoritmos

FIFO

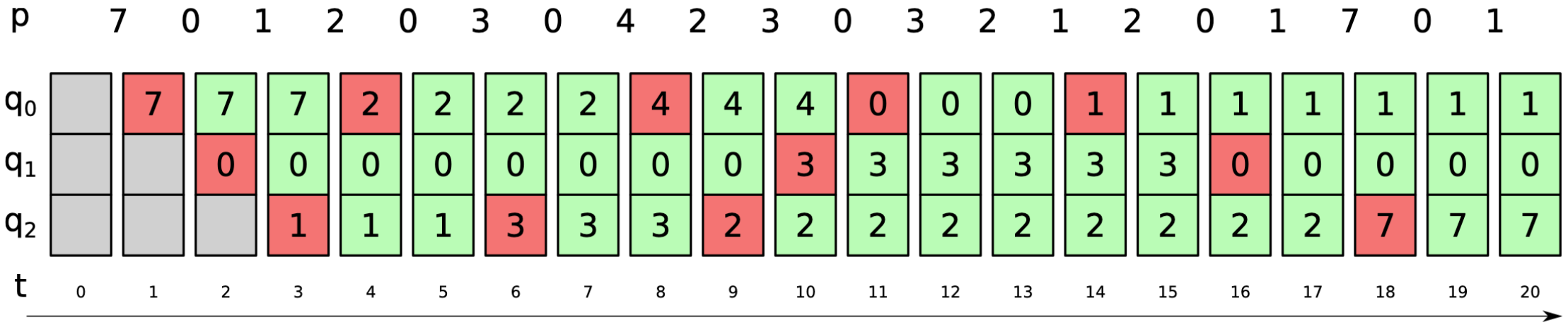
- Páginas mais antigas são removidas para dar lugar a novas páginas
- Fácil de implementar
- No exemplo: 15 *page faults*



Fonte da Imagem: Prof. Maziero

LRU

- Páginas de uso recente podem ser acessadas num futuro próximo
- Retira da RAM páginas que não foram usadas recentemente
- No exemplo: 12 *page faults*



Fonte da Imagem: Prof. Maziero

Conclusão

- **Objetivos do Gerenciamento de Memória:** Alocar e desalocar memória eficientemente, proteger a memória de processos, e fornecer um espaço de endereçamento virtual maior que a memória física disponível.
- **Particionamento:** Uma técnica básica que divide a memória em partições para alocação aos processos (particionamento fixo vs. particionamento variável). Problemas como fragmentação interna e externa são inerentes.
- **Paginação:** Divide tanto o espaço de endereçamento virtual quanto a memória física em páginas/frames, eliminando a fragmentação externa. Utiliza tabelas de páginas para mapear páginas virtuais para frames físicos.
- **Memória Virtual:** Uma técnica poderosa que permite aos processos usarem mais memória do que está fisicamente disponível, utilizando o disco como extensão da RAM.

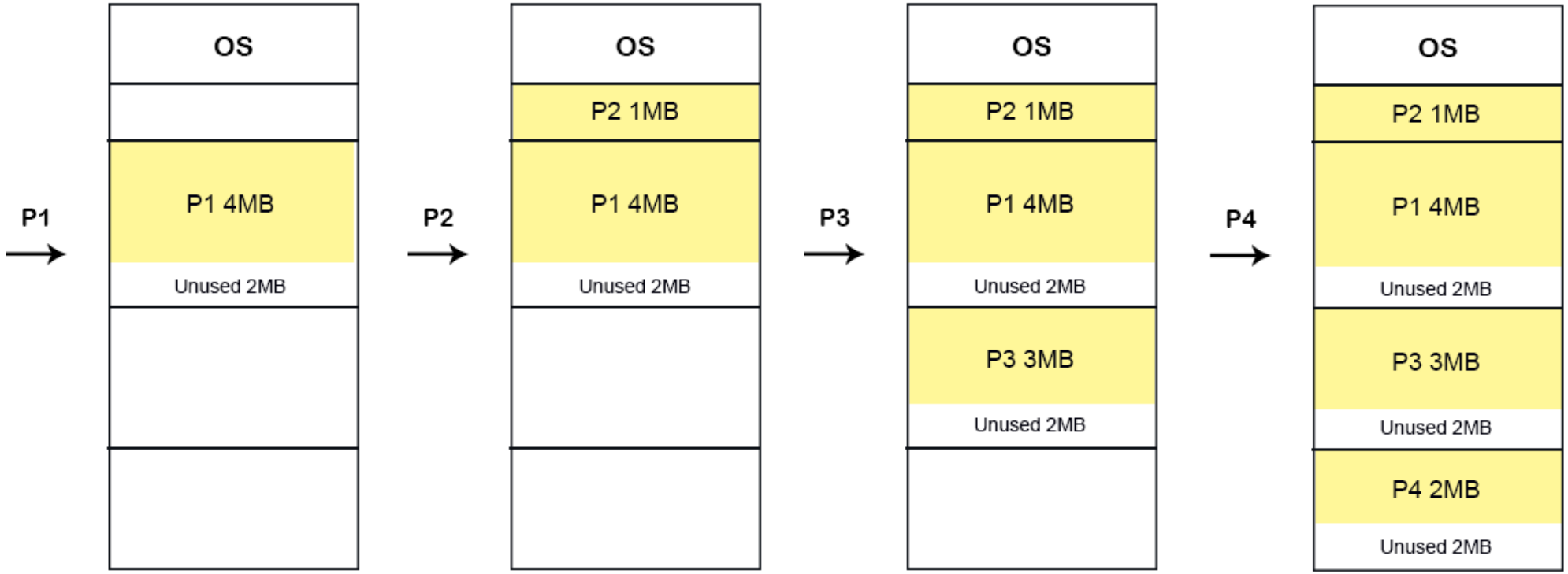
Conclusão (cont.)

- **TLB (Translation Lookaside Buffer):** Um cache de mapeamentos de páginas para acelerar a tradução de endereços virtuais em físicos.
- **Técnicas de Substituição de Páginas:** Algoritmos como FIFO, LRU e Optimal são usados para decidir quais páginas substituir quando ocorre um page fault.
- Leitura adicional:
  - Capítulos 14 a 17 do livro Sistemas Operacionais: Conceitos e Mecanismos de por Prof. Carlos A. Maziero.
  - Capítulos 8 e 9 do livro *Operating Systems Concepts* (Silberschatz)
  - As apresentações sobre Virtual Memory em IT Systems – Open Educational Resource, produzido por Jens~Lechtenböger ajudam muito a compreender melhor o tema.

Material Adicional

Estratégias de Alocação de Memória

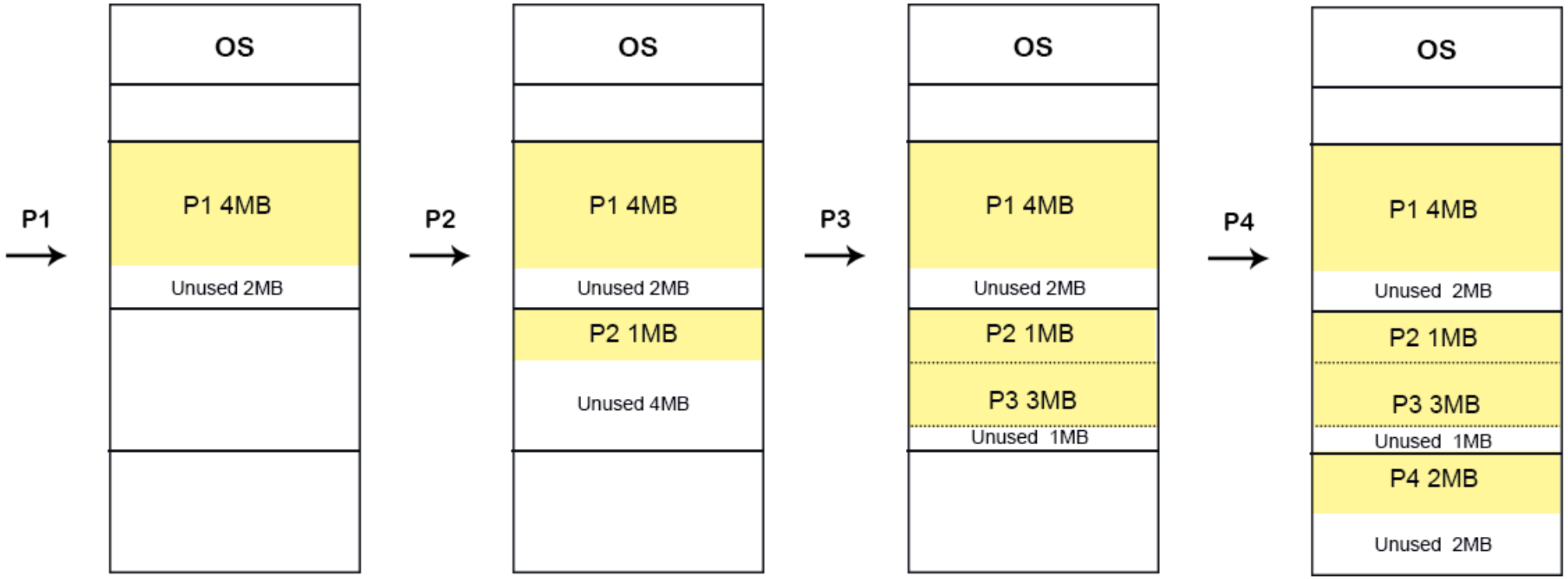
- **First Fit**: Escolhe a primeira área livre que satisfaça o pedido de alocação; tem como vantagem a rapidez, sobretudo se a lista de áreas livres for muito longa.



Fonte da Imagem: First Fit @CSTaleem

Estratégias de Alocação de Memória

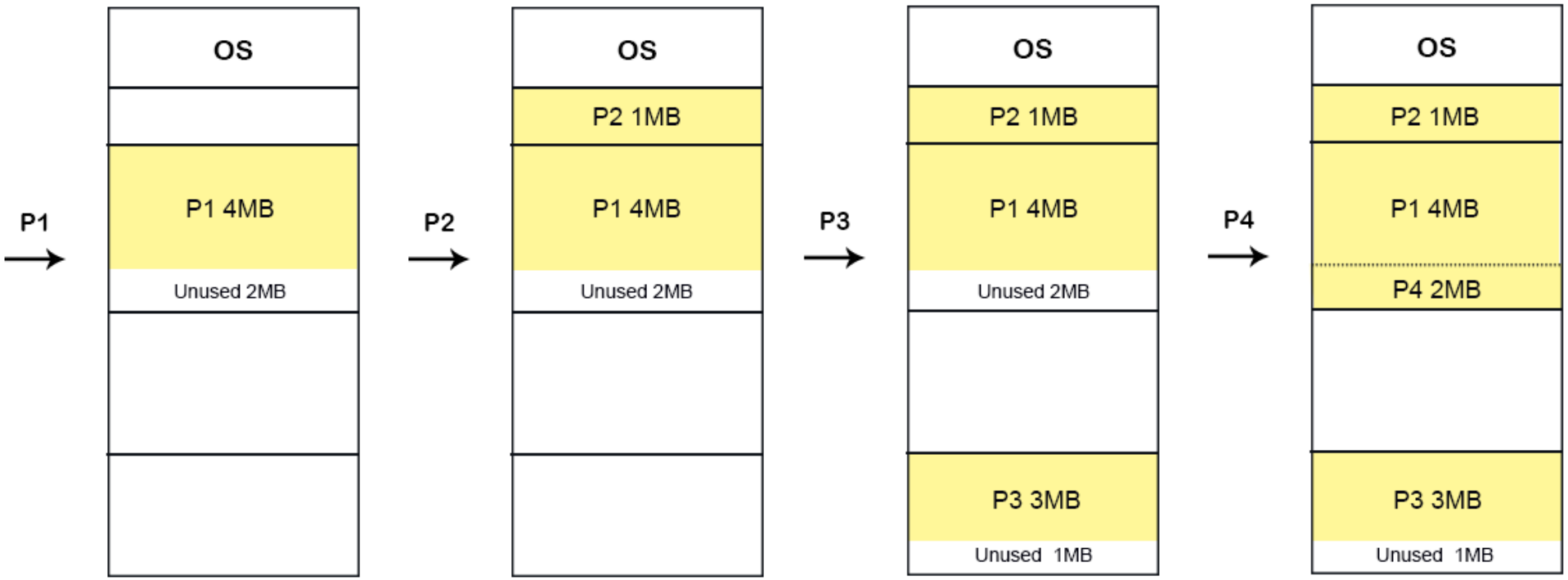
- **Next Fit**: Percorre a lista de áreas a partir da última área alocada ou liberada, para que o uso das áreas livres seja distribuído de forma mais homogênea no espaço de memória.



Fonte da Imagem: Next Fit @CSTaleem

Estratégias de Alocação de Memória

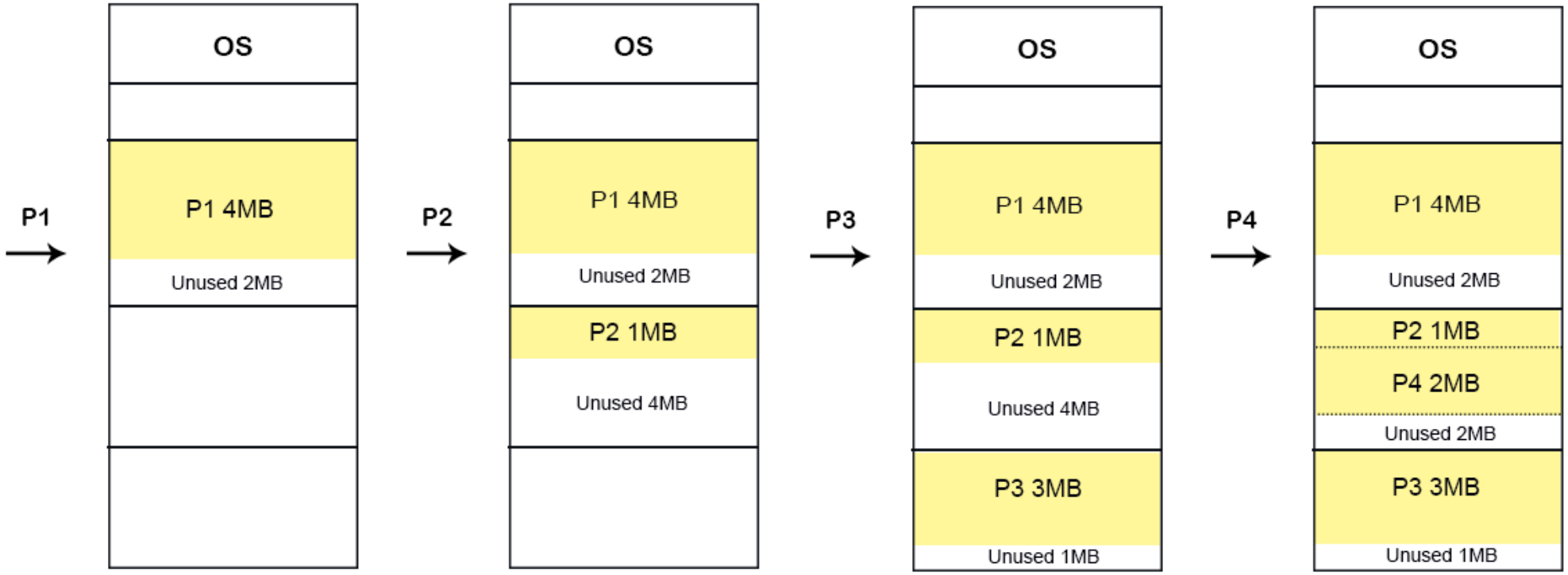
- **Best Fit**: Escolhe a menor área possível que possa receber a alocação, minimizando o desperdício de memória. Contudo, algumas áreas livres podem ficar pequenas demais e com isso se tornarem inúteis



Fonte da Imagem: Best Fit @CSTaleem

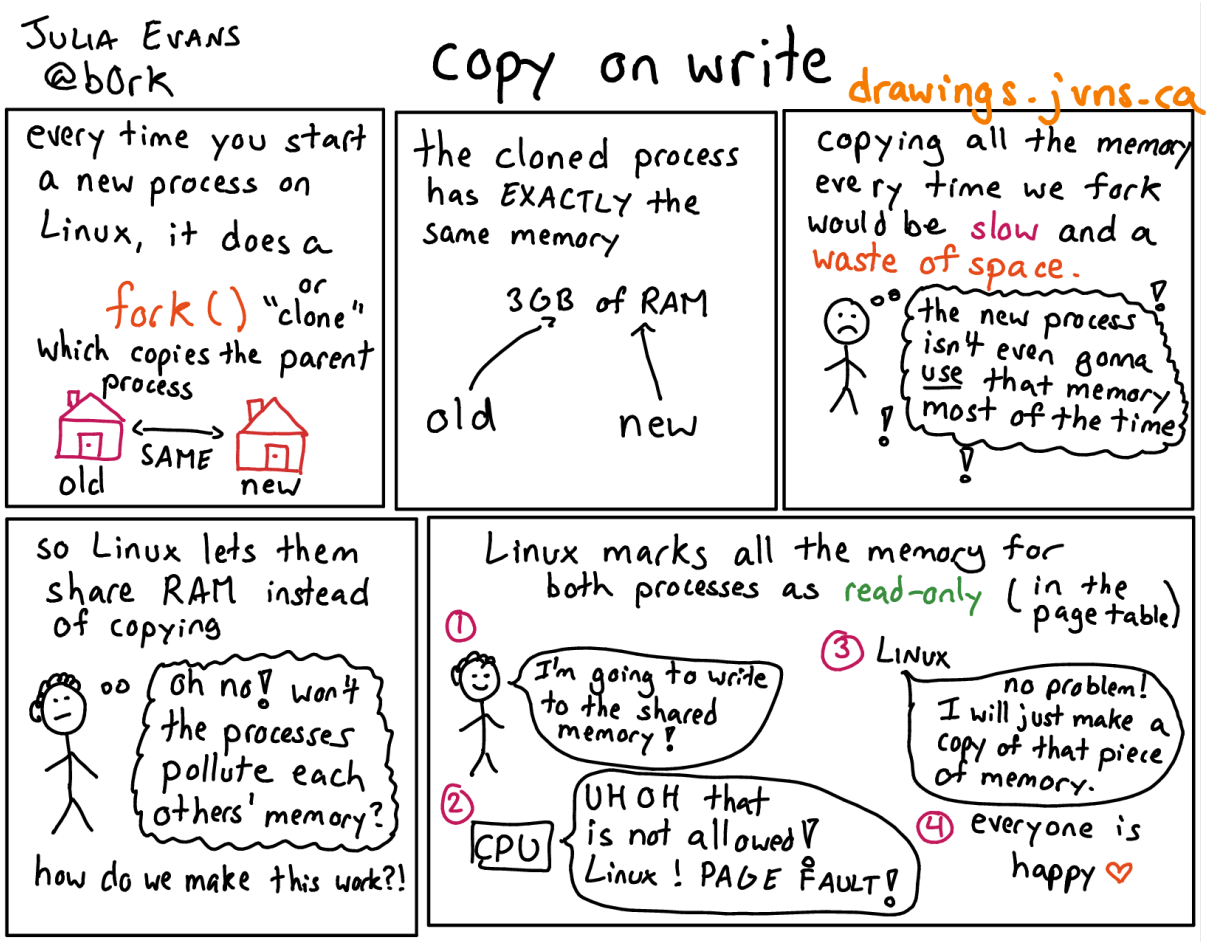
Estratégias de Alocação de Memória

- **Worst Fit**: Escolhe sempre a maior área livre possível, de forma que a "sobra" seja grande o suficiente para ser usada em outras alocações.



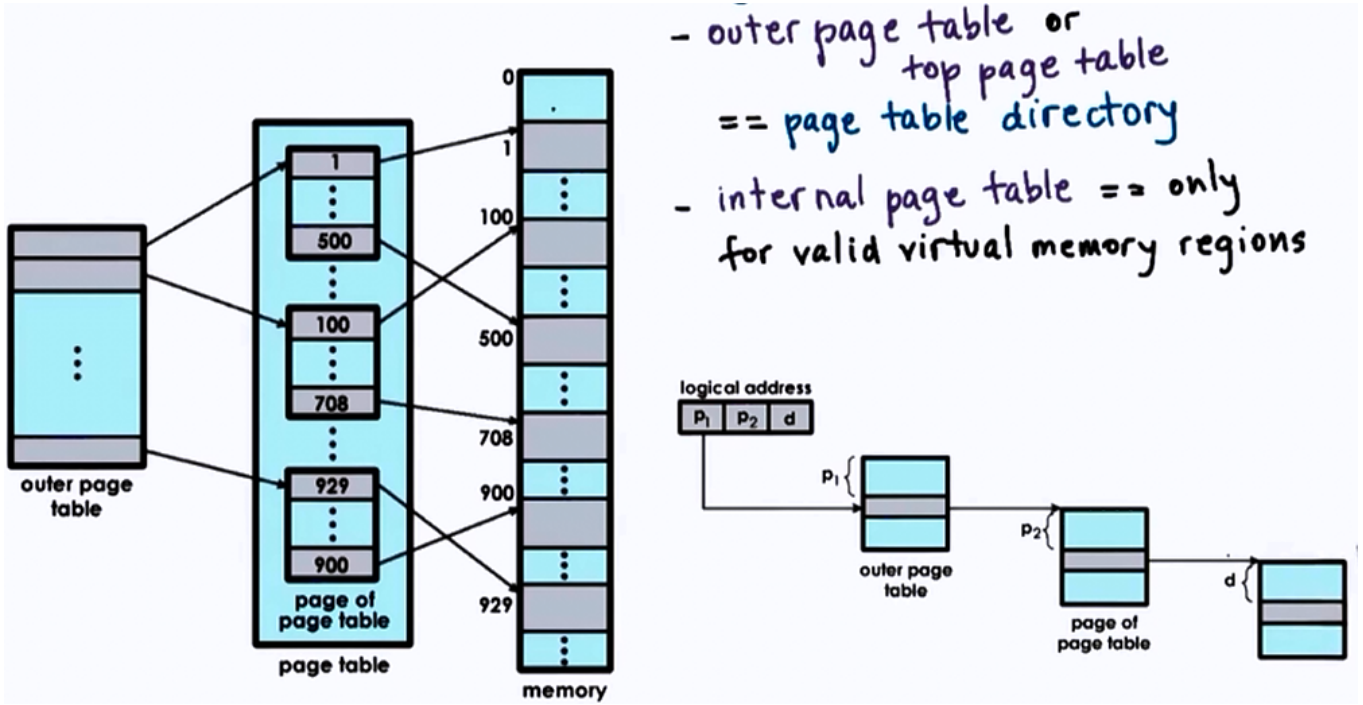
Fonte da Imagem: Worst Fit @CSTaleem

Copy On Write



Fonte da Imagem: JVNS

Tabela de Páginas Multinível



Fonte da Imagem: OS Notes

- **Problema:** Tabela de páginas pode ficar excessivamente grande.
- **Solução:** Dividir em porções menores
- Mais informações em Virtual Memory II by Jens Lechtenbörger

Vídeos

- But, what is Virtual Memory? YouTube
- Page Tables and MMU: How Virtual Memory Actually Works Behind the Scenes (Animation) YouTube
- MM101: Introduction to Linux Memory Management YouTube
- LRU Cache YouTube
- Least Recently Used (LRU) Explanation YouTube