Aula 12

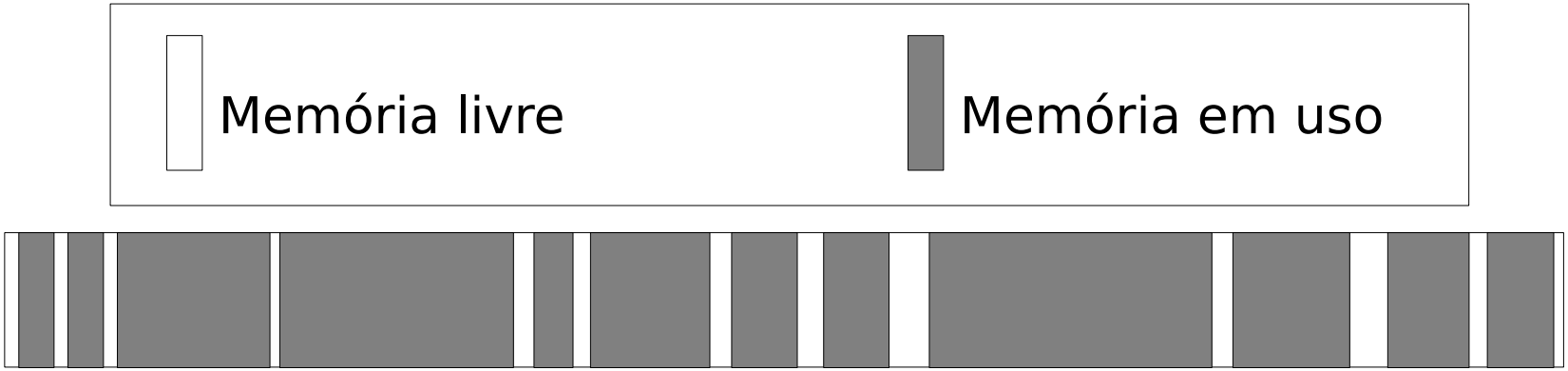
Gerência de Memória - Paginação

1.1 Gerência de memória- o contexto da paginação
1.2 O que é a paginação
1.3 Implementação
1.4 Referências: Capítulo 9 (9.4)

O contexto da paginação

Revisão de gerência de memória:

Alocação dinâmica
+
Blocos de memória contíguos
=
FRAGMENTAÇÃO



O que é paginação (1)

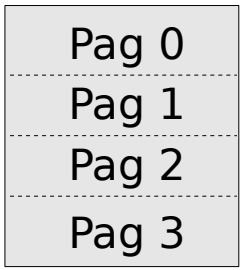
Uma solução para a fragmentação:

- “Quebrar” a memória do processo, permitindo espaços de endereçamento não contíguos
- Dividir a memória principal _E_ a memória usada pelos processos em **páginas** de, por exemplo 4K
- Carregar a memória do processo em páginas disponíveis da memória principal

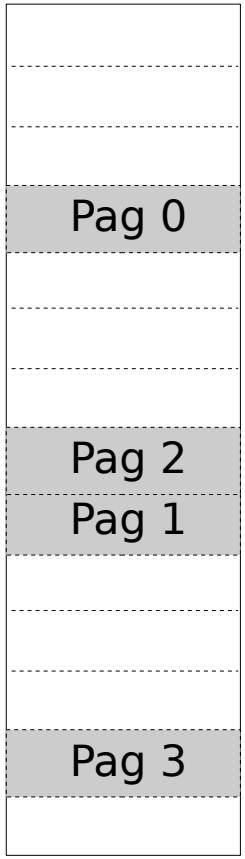
O que é paginação (2)

Uma visão simplificada:

No processo está assim



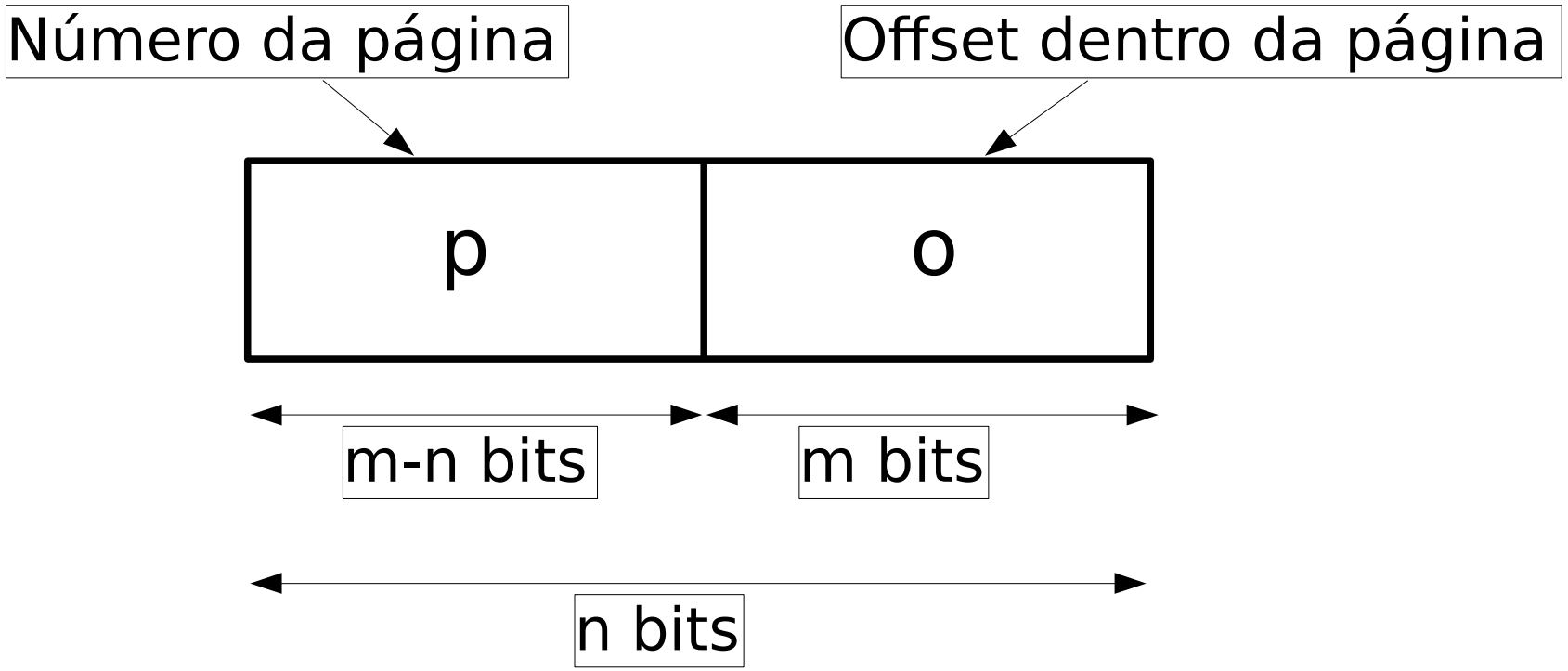
Na memória principal poderia ser assim



Vamos seguir adiante para ver como isto funciona!

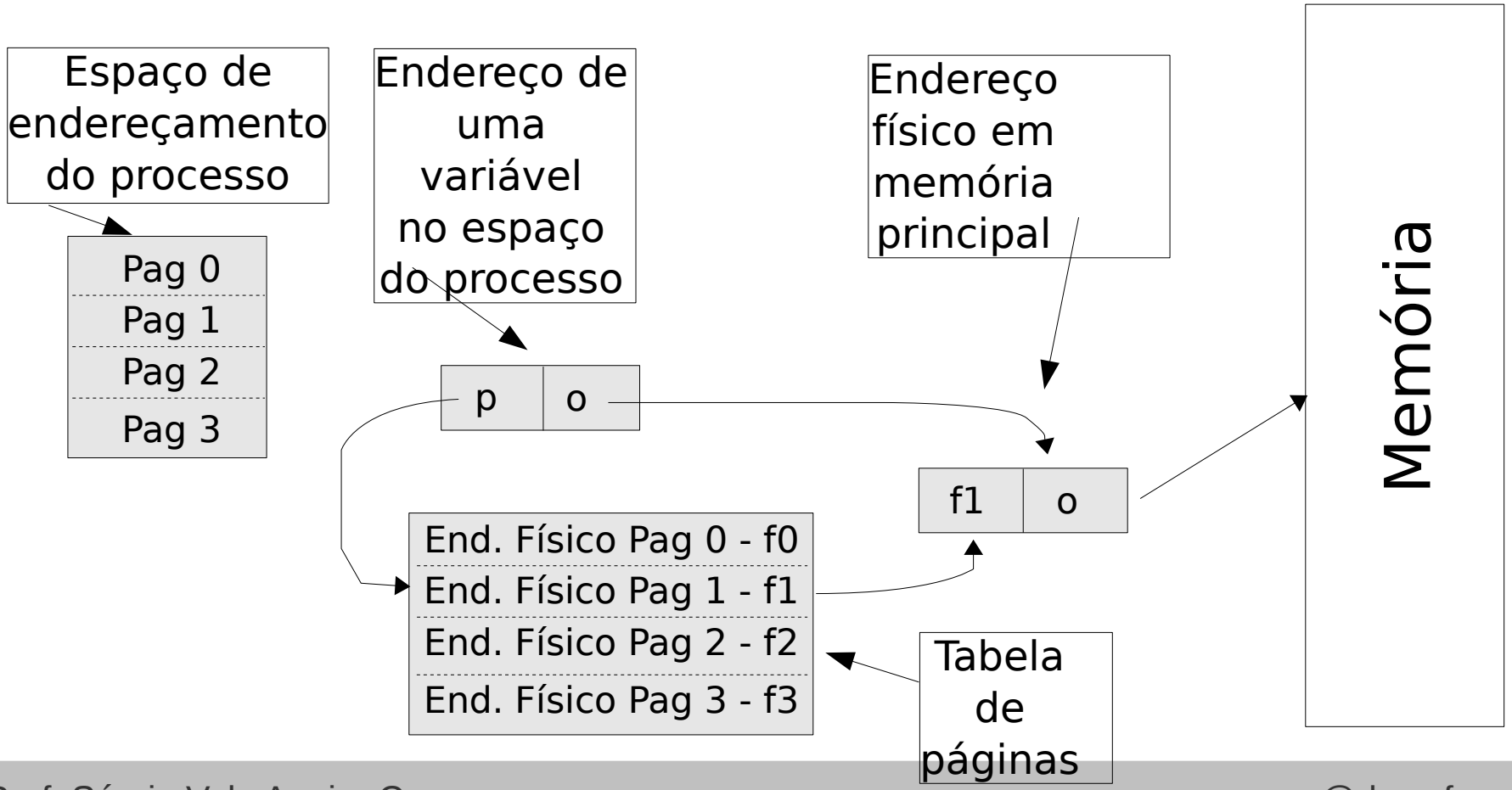
Endereçamento (1)

Todo endereço é dividido em duas partes:



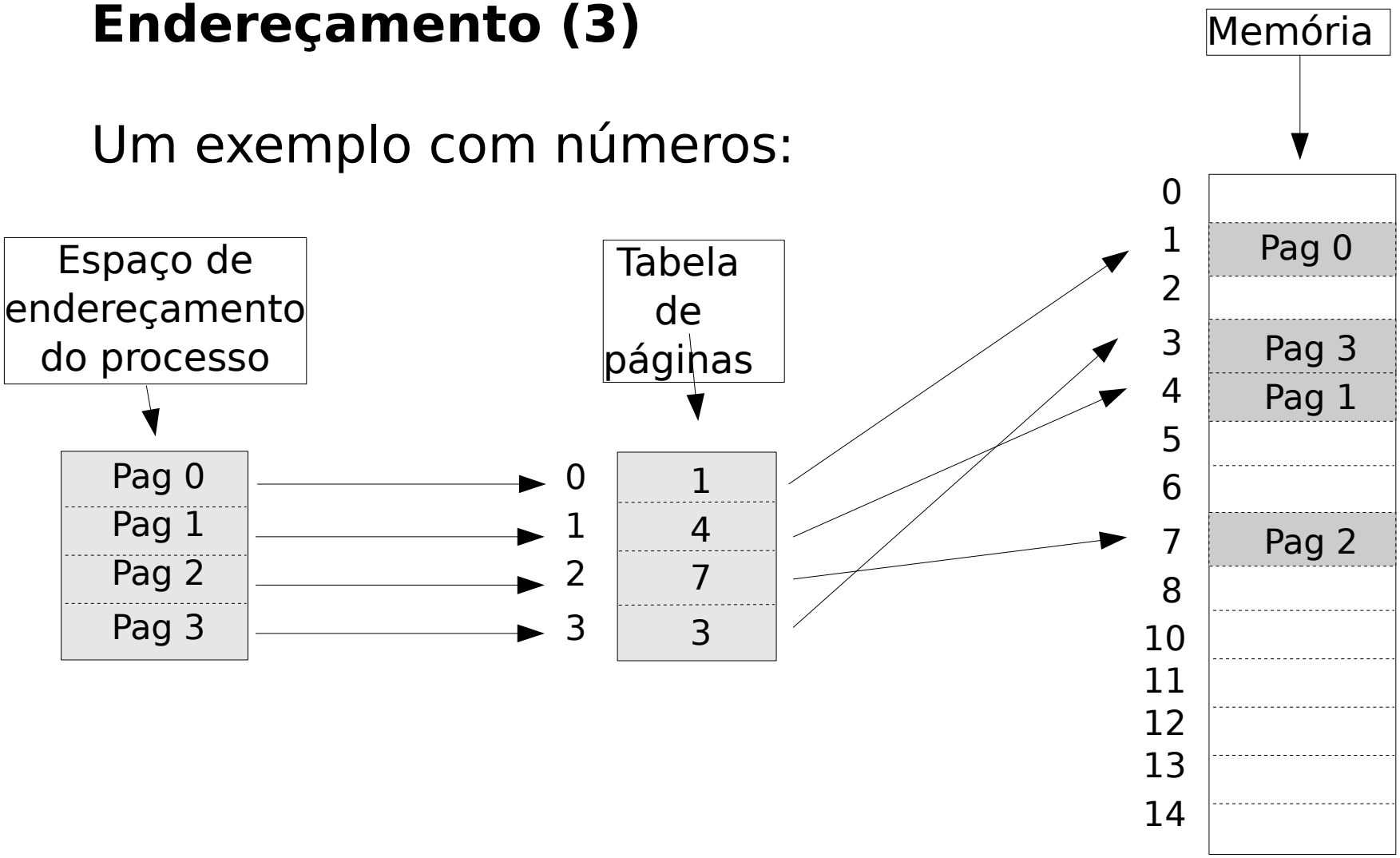
Endereçamento (2)

Achando o endereço físico de uma variável:



Endereçamento (3)

Um exemplo com números:

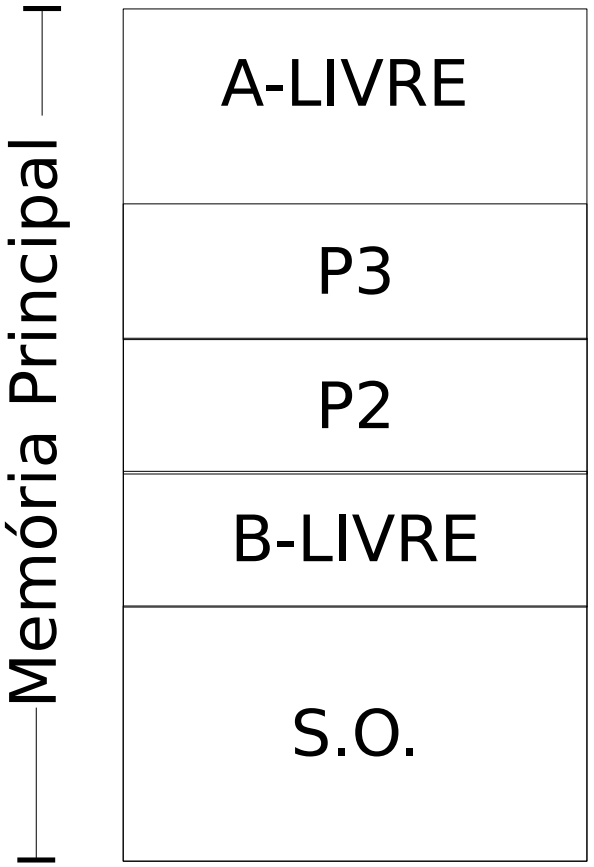


Fragmentação com paginação (1)

A paginação resolve a fragmentação?

- **Fragmentação externa**:
  - Entre processos
  - Ao nível do S.O.

RESOLVIDO

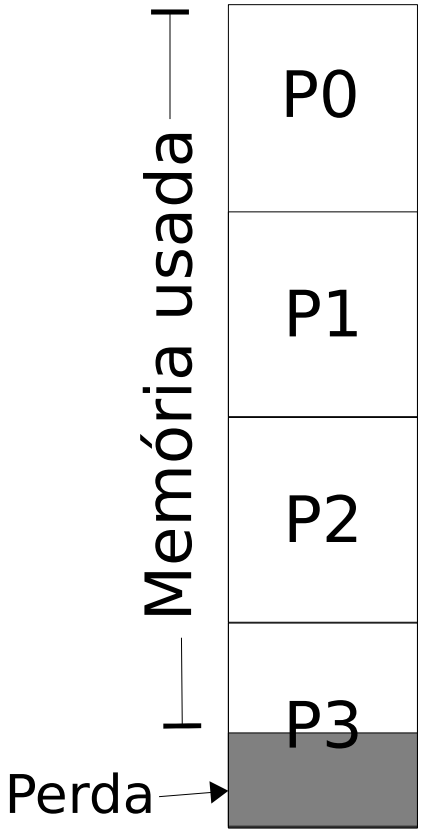


Fragmentação com paginação (2)

A paginação resolve a fragmentação?

- **Fragmentação interna**:
  - Na média, a última meia página do processo fica vazia
  - Páginas menores diminuem fragmentação...
  - ...mas aumentam a tabela de páginas
  - Tipicamente páginas têm entre 2 a 8 K de tamanho

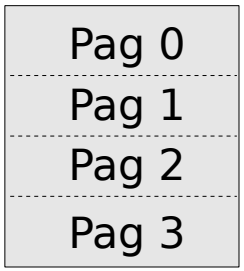
PROBLEMA



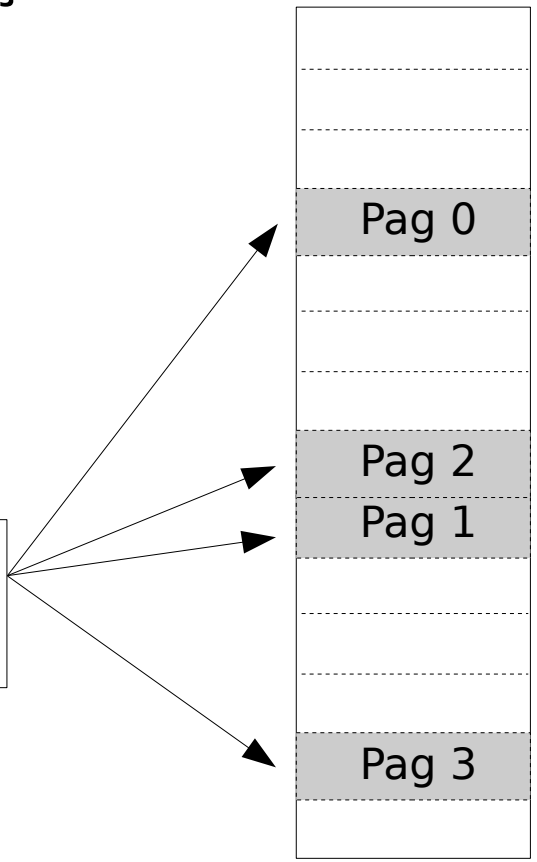
Paginação e separação de visões

Um aspecto importante da paginação é a separação de visões:

Um **processo** "vê" sua memória assim: um espaço de endereçamento contínuo



E o **S.O.** vê assim: páginas espalhadas e não contínuas

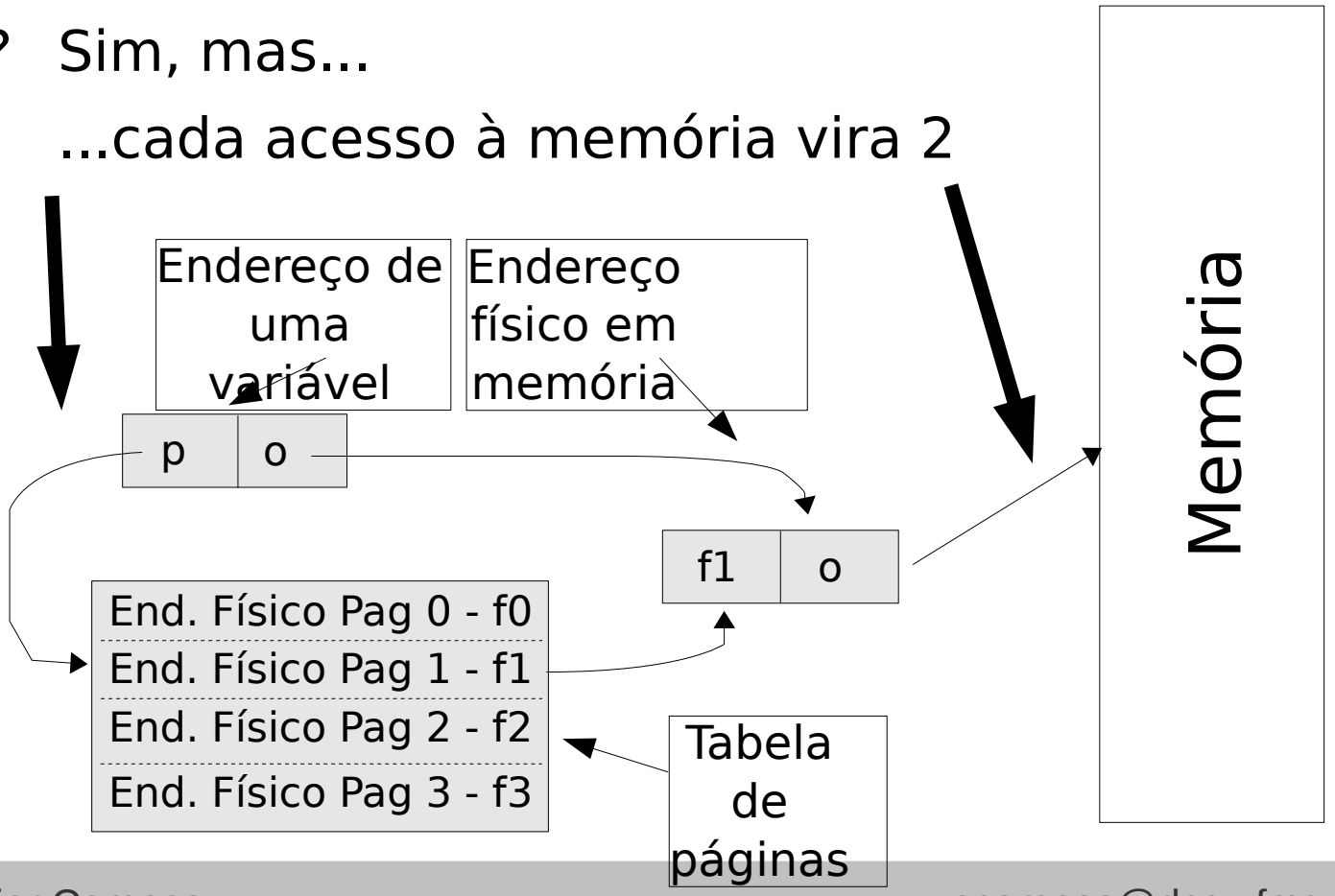


Implementação (1)

1a tentativa: colocar a tabela de páginas na memória

Funciona? Sim, mas...

...cada acesso à memória vira 2



Implementação (2)

2a tentativa: colocar a tabela de páginas em registradores

Funciona? Sim, mas...

...fica **MUITO** caro!

3a tentativa: Recorrer ao velho e bondoso *cache*...

Vejamos a seguir!

Translation Look-Aside Buffer (TLB)

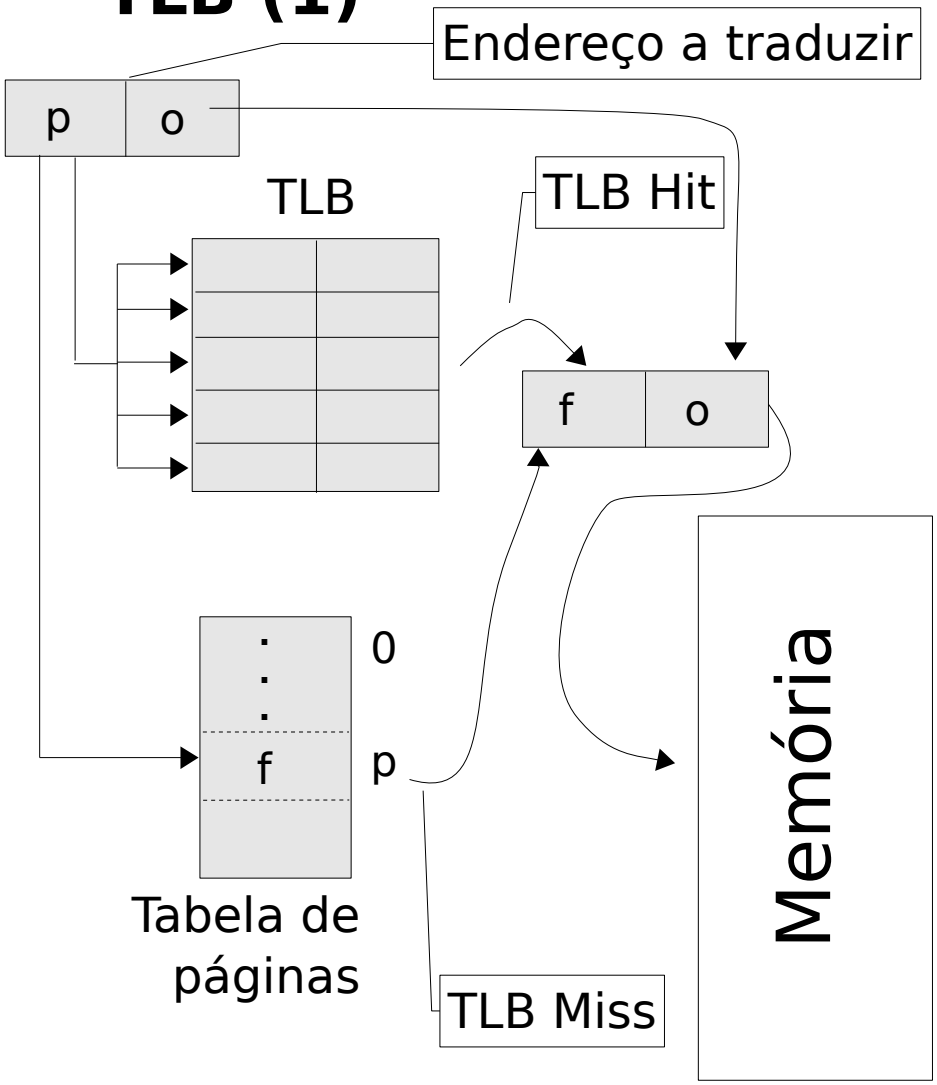
Um conjunto de registradores especiais:

- Super rápido (são registradores)
- Cada registrador tem duas partes:
  - chave e valor
- Dada a chave, retorna o valor correspondente
- A busca é feita em todos os registradores **simultâneamente**
- É uma espécie de *cache* da tabela de páginas

Porquê é mais barato que colocar a tabela de páginas em registradores?

- Porquê é um *cache*. Armazena parte da tabela!

TLB (1)



O número de página é olhado no TBL;

Se achar, **TLB HIT** e retorna a página física

Se não achar, **TLB MISS:**

- Busca número da página na tabela de páginas (memória principal)
- Escolhe e limpa uma entrada no TLB
- Guarda o par (pag. Lógica e pág. física no TLB)
- Retorna a página física

TLB (2)

Pense... o acontece quando ocorre troca de contexto?

- Um processo “perde” o processador...
- ...outro “ganha” o processador
- Como fica o TLB?

O TLB é descarregado quando ocorre troca de contexto!

Tempo de acesso (1)

Suponha que se gaste 20ns para olhar o TLB e 100ns para ler da memória:

- TLB hit gasta 20 + 100 = 120ns
- TLB miss gasta: 20 + 100 + 100 = 220ns

O **hit rate** diz a frequência com que achamos a página desejada no TLB. Com um hit rate de 80%, o tempo de acesso será:

0.8 * 120 + 0.2 * 220 = 140ns

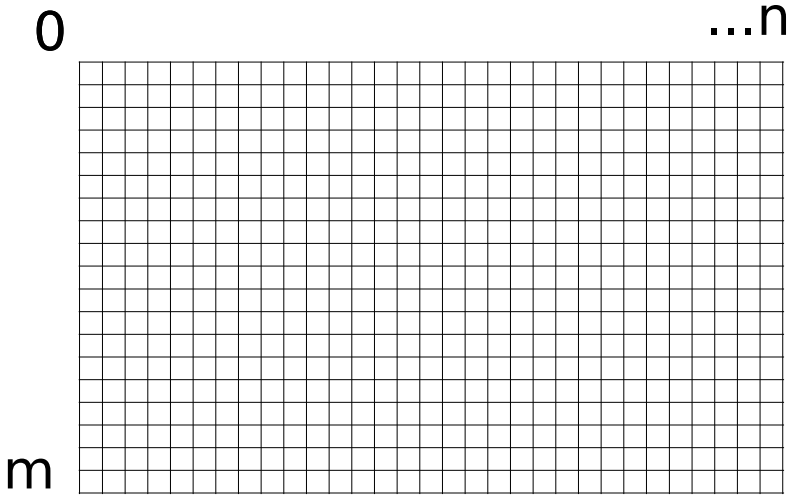
Programa inimigo do TLB

Do que depende o hit rate:

- Do tamanho do TLB
- Do padrão de acesso do programa

O 486 tinha 32 posições no TLB e hit rate de 98%
Será que este hit rate vale para qualquer programa? Vejamos a seguir

Tipos de memória (2)



- Matriz armazenada por linha
- Cada linha em uma página
- Ocupa 4meg de memória

Como varrer a matriz?

- Por linha: 1 TLB miss por linha, *m* no total
- Por coluna: *m* TLB misses por coluna, *m***n* no total!

Paginação multinível(1)

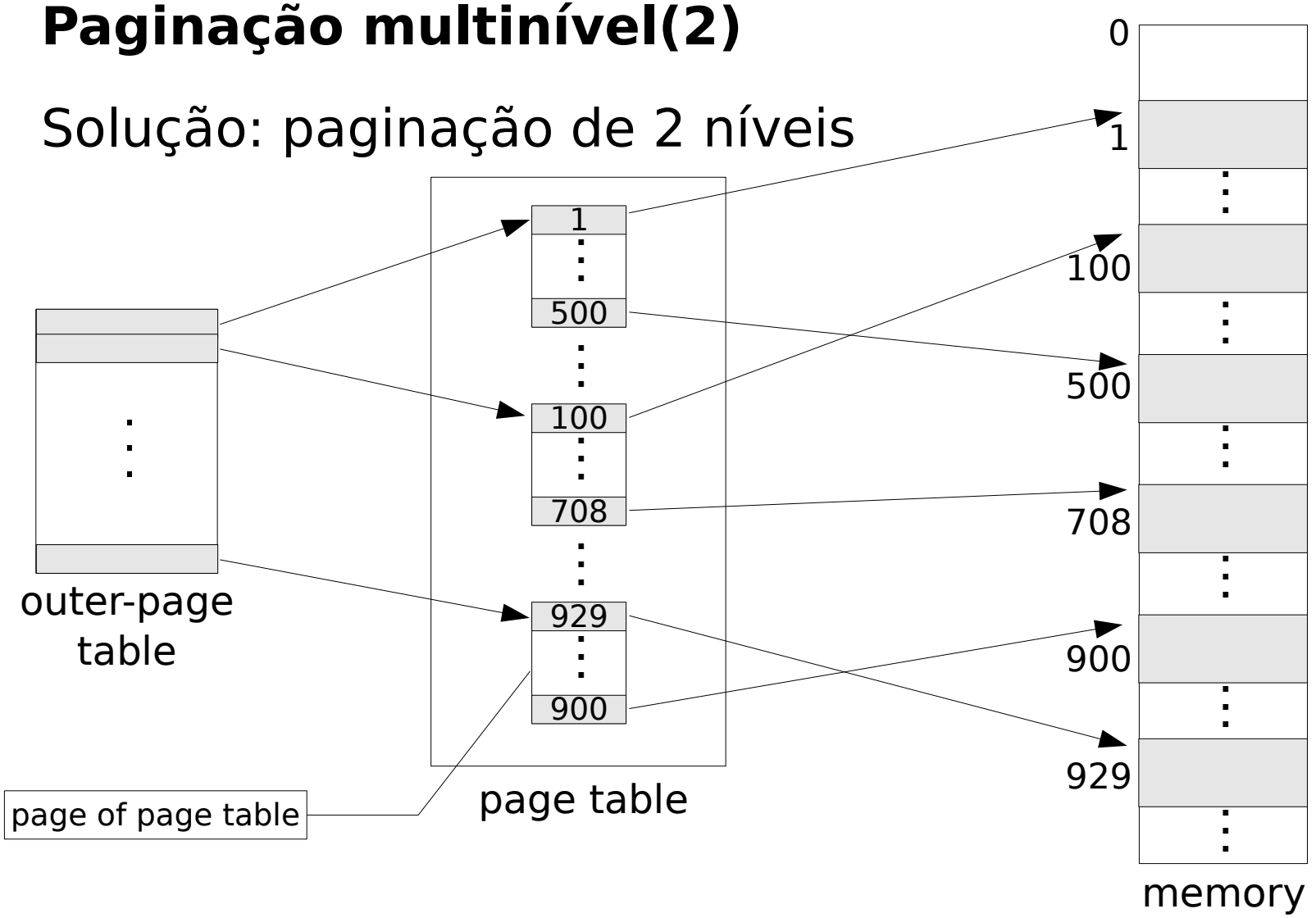
Espaços de endereçamento são muito grandes.

Exemplo 1: processador de 32 bits, página de 4K:
Tabela de páginas: 1 milhão de páginas, 4 megs!

Exemplo 2: processador de 64 bits, página de 4K:
Tabela de páginas.... 2^64 / 2^12
2^52 páginas!

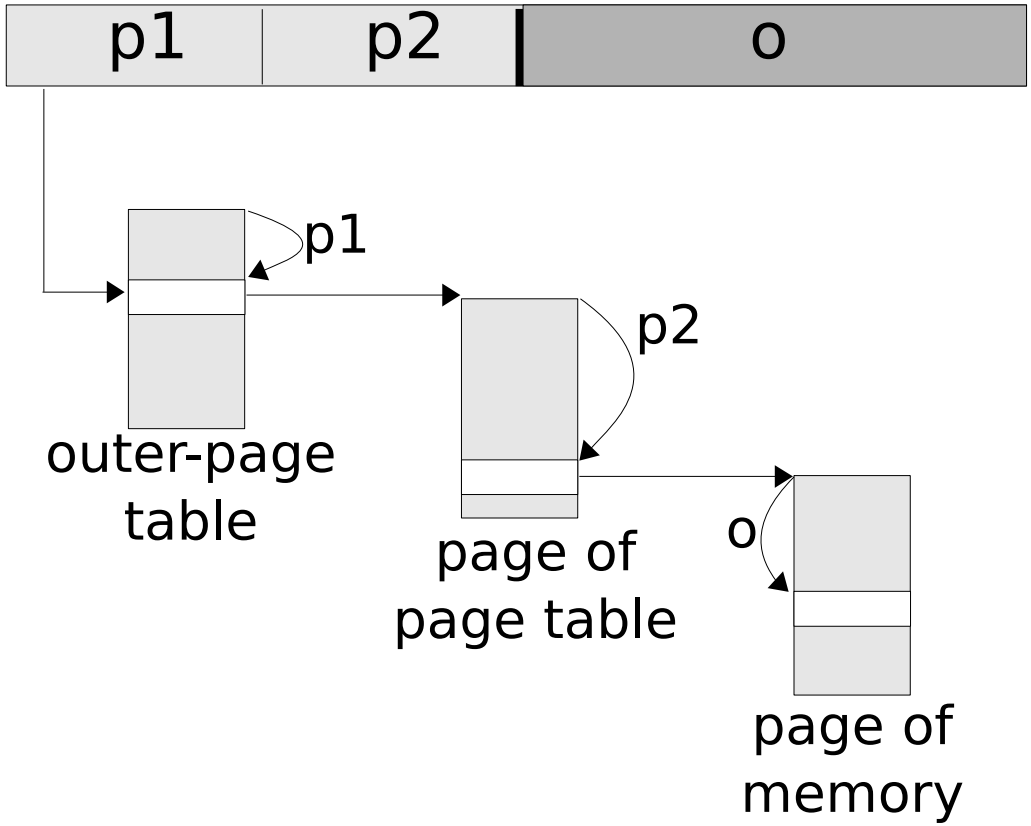
Paginação multinível(2)

Solução: paginação de 2 níveis



Paginação multinível(3)

Solução: paginação de 2 níveis



Tempo de acesso

Suponha que:

- TLB hit gasta 20 + 100 = 120ns
- Hit rate de 98%

Com um nível de paginação:

- TLB miss gasta 20 + 100 + 100 = 220ns
- Tempo de acesso: .98*120 + .02*220 = 122ns

Com dois níveis de paginação:

- Tempo de acesso: .98*120 + .02*320 = 124ns

Com três níveis de paginação:

- Tempo de acesso: .98*120 + .02*320 = 126ns

Conclusão: Desempenho não sofre!

Localidade e tempo de acesso

Mas, como assim, não sofre?

Não sofre, desde que se mantenha a **localidade**!

- programas acessam dados próximos em intervalos de tempo pequenos
- lembram-se do progama de matrizes?

Conclusão: **localidade** é um conceito extremamente importante para o **desempenho**

Conclusão

Paginação: técnica para resolver a fragmentação externa – problema mais crítico que a fragmentação interna

TLB – translation look-aside buffer: cache, em registrador, para tornar a pesquisa na tabela de páginas eficiente

Paginação multinível: a própria tabela de páginas é paginada, porque o espaço de endereçamento é muito grande

Localidade: fator crucial para o desempenho