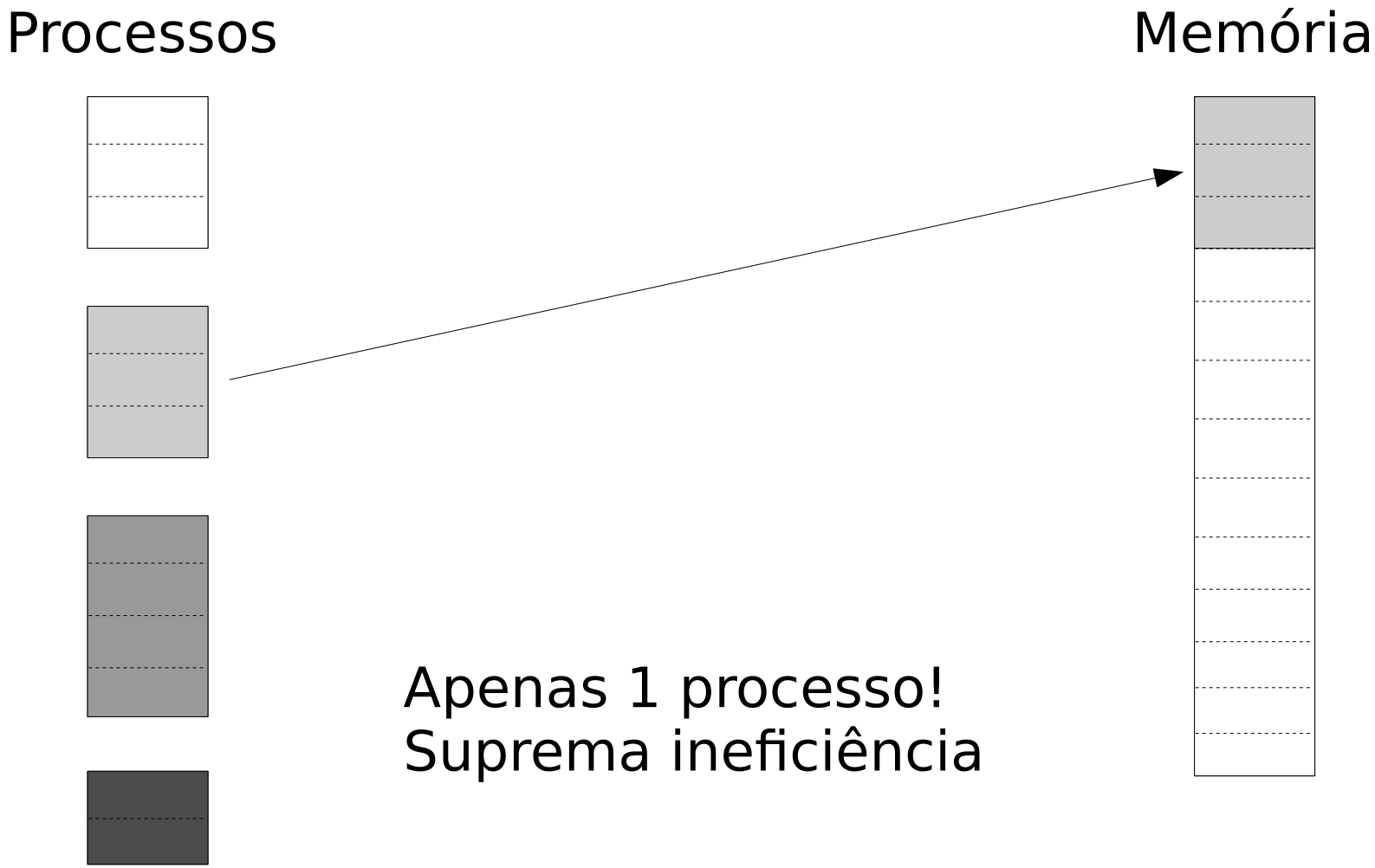
Aula 14

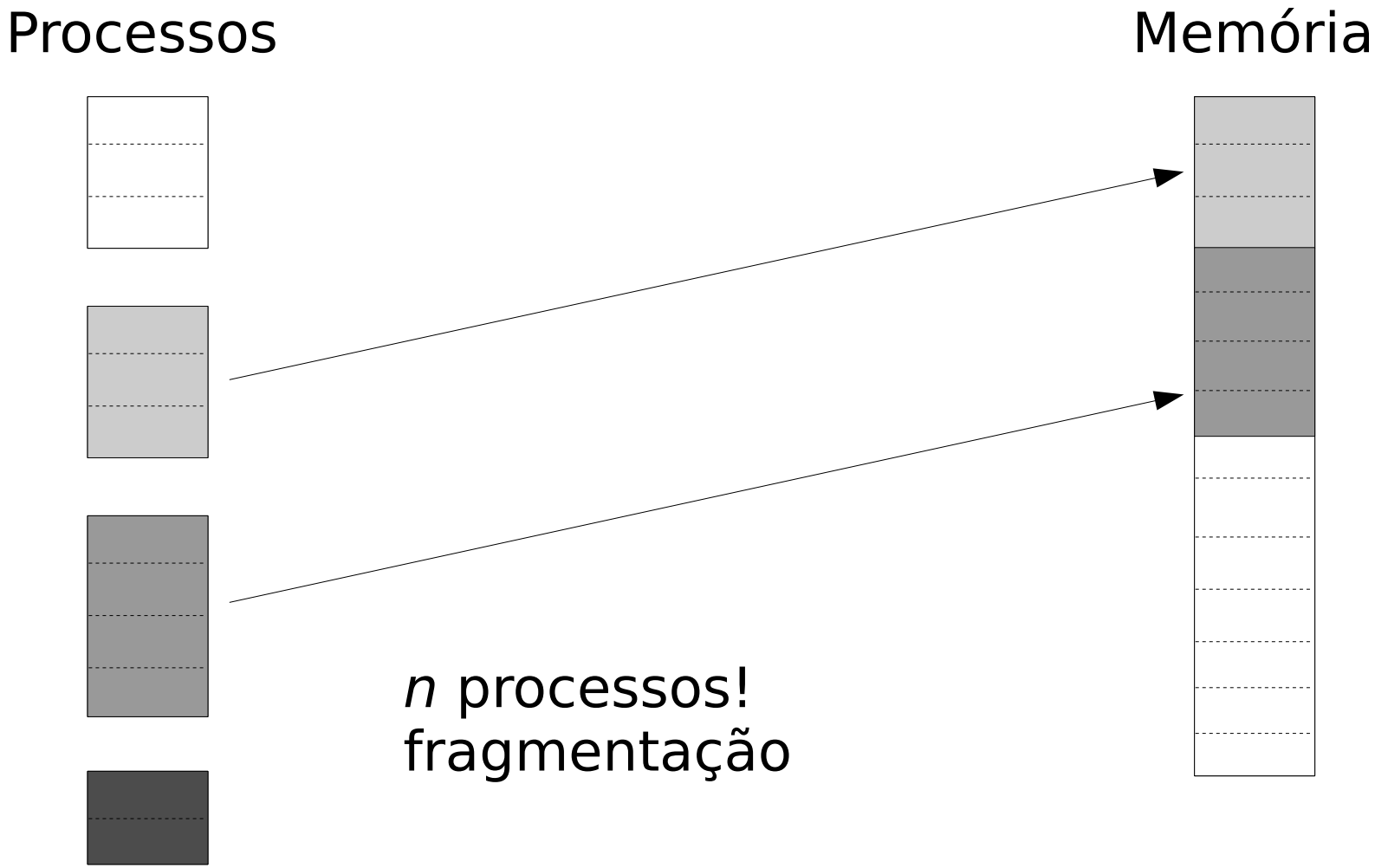
Memória Virtual

1.1 Linha evolucionária
1.2 O que é
1.3 Implementação
1.4 Problemas

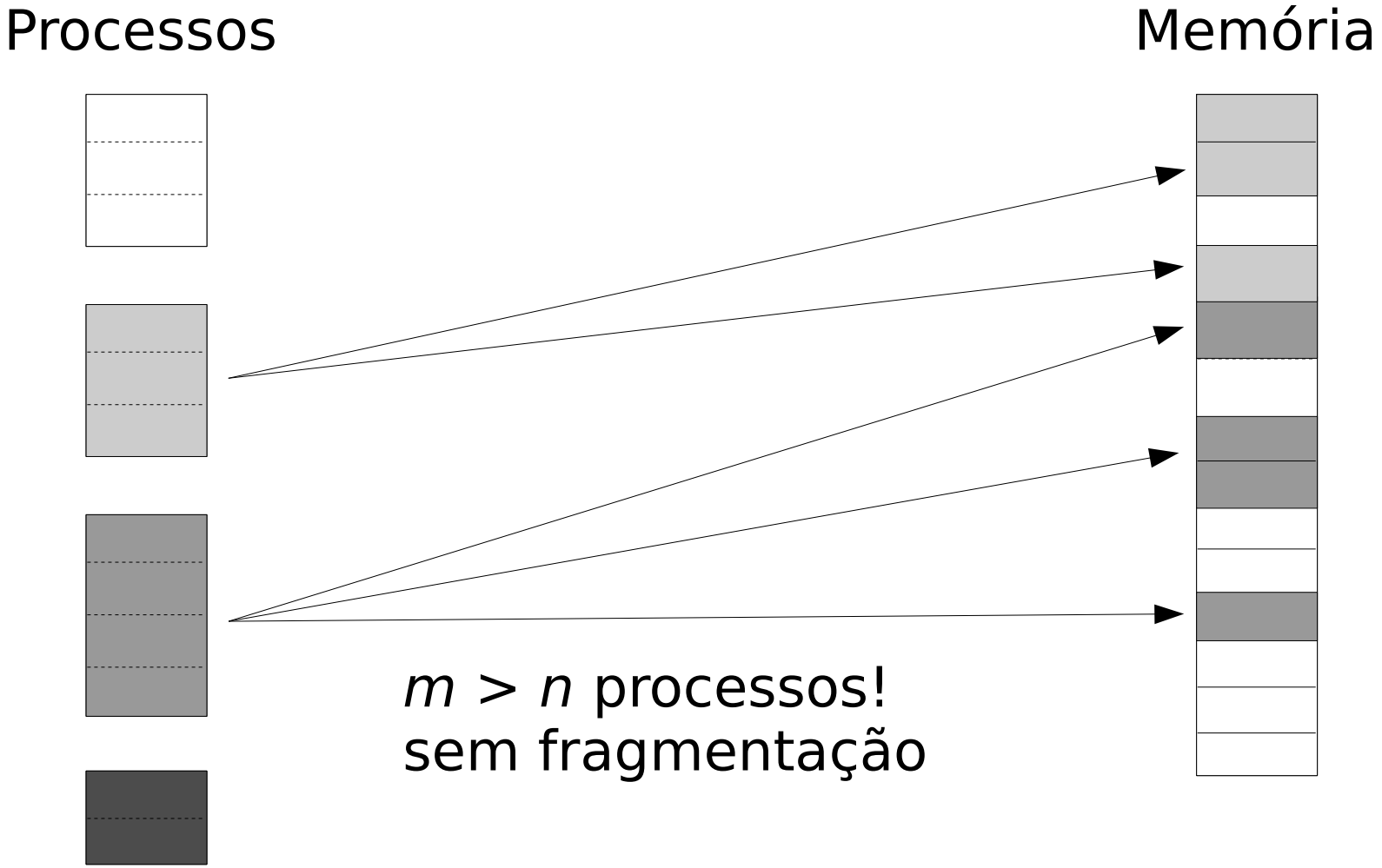
A Idade da Pedra Lascada



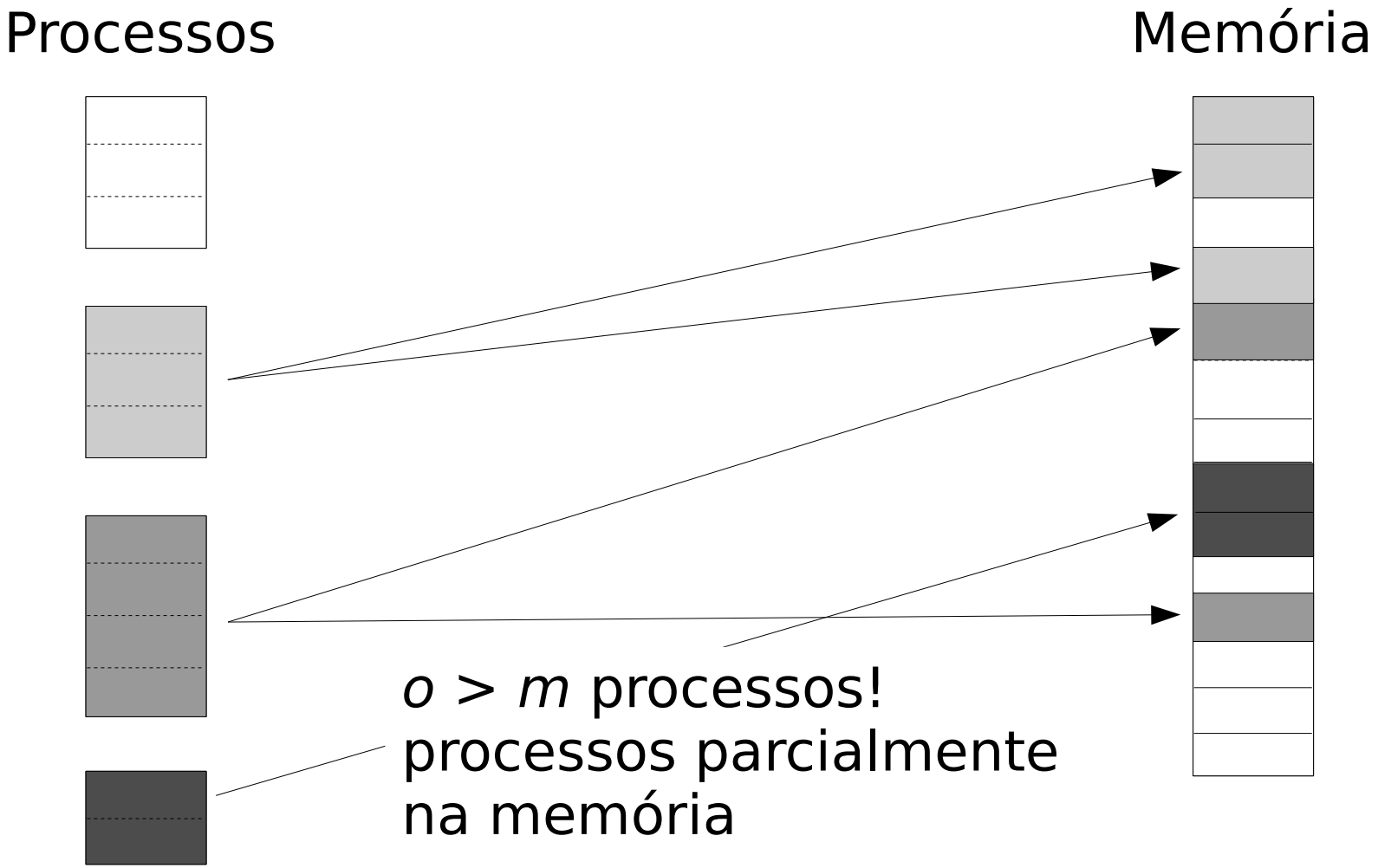
A Idade da Pedra Polida



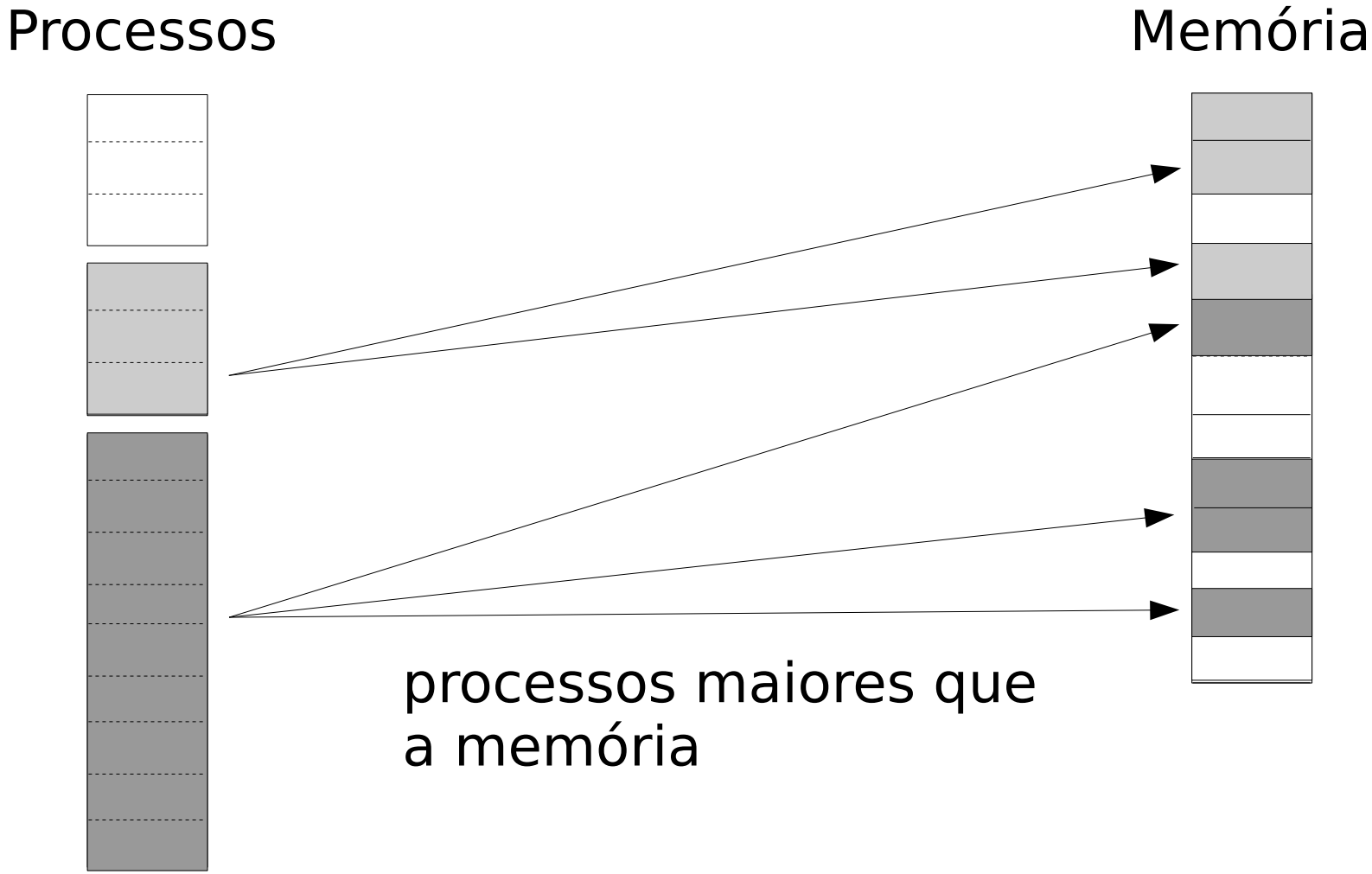
A Idade Média



A Idade Virtual



A Idade Virtual - 2



Memória Virtual - 1

Possibilita a execução de processos maiores que a memória:

- Portabilidade:
  - Programas podem ser escritos sem limitações de de memória disponível
  - A quantidade de memória disponível afeta somente o desempenho, não o programa

- Nem sempre a execução usa todo o espaço de endereçamento todas as vezes
  - Programas como o Adobe Photoshop têm centenas de subrotinas (ex: uma para dada filtro)
  - Mas, raramente, todas são usadas em todas as execuções

Memória Virtual - 2

- Mais rápido para executar – não precisa ler todo o código e dados:
  - Imaginem carregar **todos** os filtros do Photoshop antes de executar!
- Mais processos podem executar ao mesmo tempo
- Mudança de paradigma: programas podem assumir espaço de endereçamento infinito:
  - Quer ler um arquivo enorme?
    - Mapeie o conteúdo do arquivo na memória virtual

Paginação sob Demanda

Um **paginador preguiçoso** é usado para implementar
memória virtual:

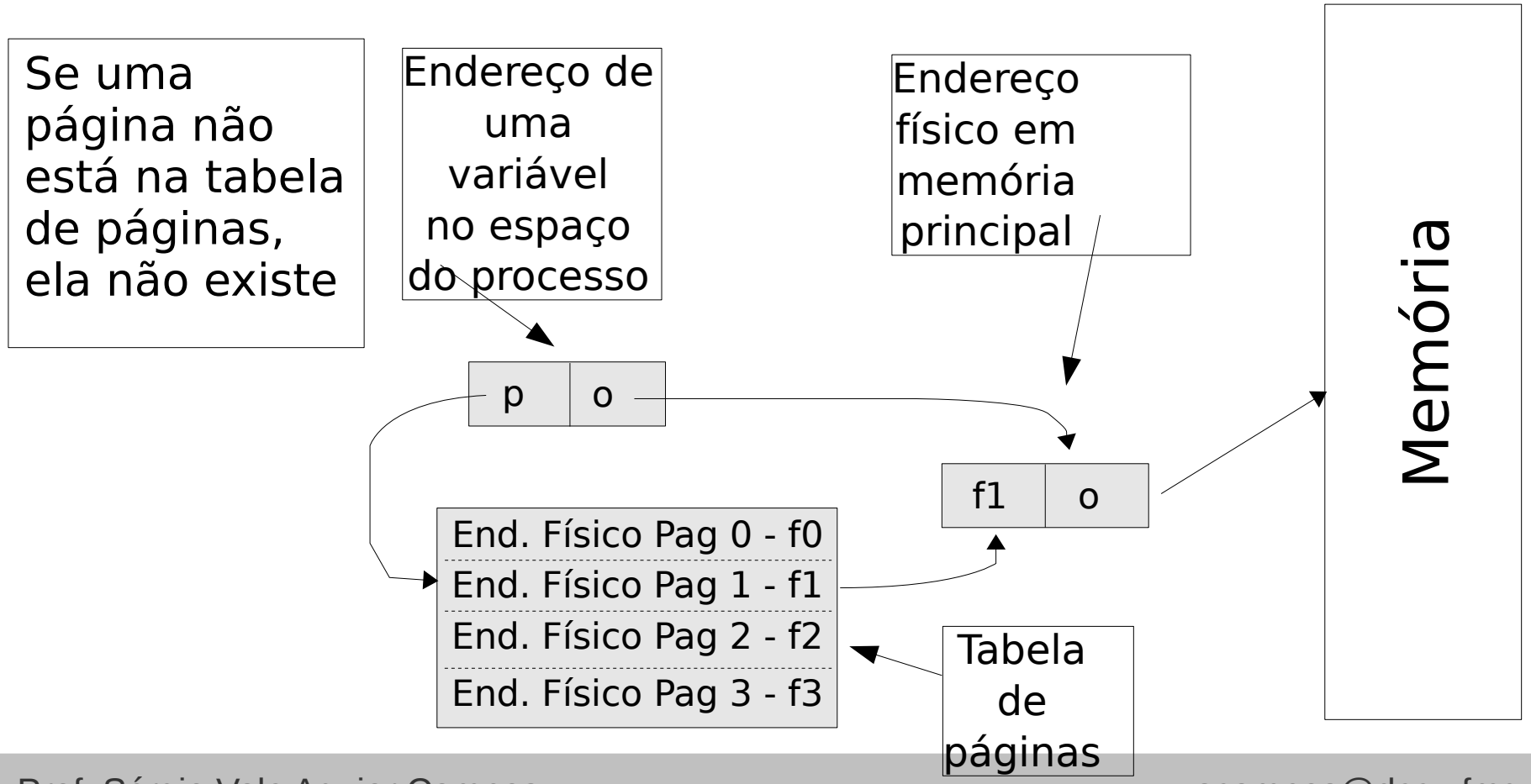
- Na hora de executar o processo o paginador carrega somente as páginas necessárias agora

Problema: O que acontece quando o programa acessa uma página que não está na memória?

- Com paginação, isto significa que a página é inválida
- Com memória virtual, isto pode significar que a página não está na memória

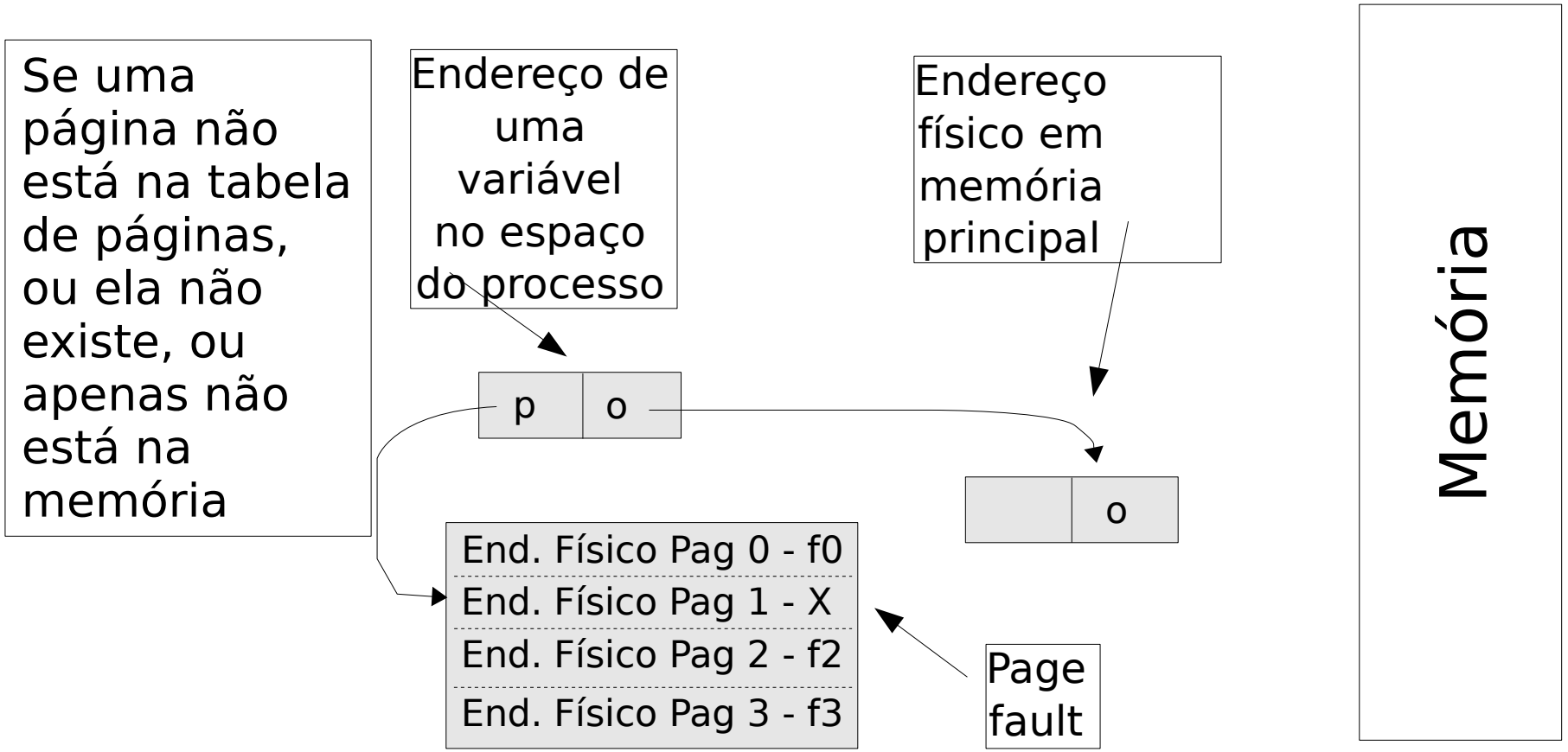
Page Fault - 1

Acesso com paginação



Page Fault - 2

Acesso com memória virtual



Page Fault - 3

Acrescentar um bit válido/inválido à tabela de páginas

Quando o acesso for feito à página inválida, gerar uma interrupção de **page fault**

Page fault:

1. Libera uma página de memória principal
2. Lê a página do disco e a coloca no lugar da página retirada
3. Atualiza a tabela de páginas
4. Retorna o processamento ao processo

Page Fault - 4

Quando o processo voltar a executar, a página estará na memória e a tabela de páginas correta

Page faults são transparentes ao processo:

- Quando ocorrem são interrupções
- Quanto terminam, o processo continua sem problemas

Desta forma, o programa não precisa estar ciente da existência de page faults, somente de um espaço de endereçamento muito grande

Problemas com a memória virtual - 1

As vantagens são muitas, mas e as desvantagens?

A implementação é complexa:
- page faults podem acontecer a qualquer momento
- têm que ser absolutamente transparentes
- confundem o pipeline do processador

Desempenho: tempo de execução pode variar muito de execução para execução...
... ou até pode até mesmo ter resultados diferentes!

Ex: e se o disco não conseguir ler a página na próxima vez?

Problemas com a memória virtual - 2

Qual página tirar quando acontecer um page fault?
- A escolha afeta o desempenho do sistema

E se a rotina de page fault não estiver na memória????
- Deadlock!

Arquitetura do processador fica muito mais complicada:

ADD (R1),(R2),(R2)

- Se de um page fault no endereço (R3), o processador tem que gerar uma interrupção no **meio** da instrução!

Problemas com a memória virtual - 3

ADD (R1),(R2),(R2)

- E se der page fault nas três página acessadas?

- Felizmente, a localidade faz que isto ocorra raramente

Preço de Page Faults - 1

A. Acesso à página na memória 126ns. Média de:
- TLB hit: 120ns
- TLB miss: 420ns

B. Acesso à página de disco:
- 420ns +
- 8ms (latência de disco)
- 15ms (seek)
- 1 ms (transferência de dados)

ou seja, uns 25 ms

Preço de Page Faults - 2

Tempo de acesso é (p é a probabilidade de page fault):

$(1-p)\ 126 + p * 25.000.000$

Agora você sabe porque é melhor dobrar a memória que trocar um processador por um duas vezes mais rápido.

Swap Space - 1

Aonde guardar as páginas descartadas da memória?

- No arquivo de onde vieram
- E se elas contiverem variáveis dinâmicas?

Solução: reservar espaço em disco, como *swap*, i.e., como armzenamento temporário

Swap é mais eficiente que arquivos em geral:

- Alocação pode ser em blocos maiores
- Mais fácil achar dados, não tem que procurar o arquivo no diretório

Swap Space - 2

Quando usar *swap* e quando usar arquivo?

1. Swap só quando o arquivo não existe

2. Ler o arquivo e gravar em swap:
   - No final, é necessário *flush* doswap

3. Copiar o arquivo inteiro para swap

Solução 2 é um compromisso razoável. É usada no BSD Unix

Swap Space - 3

Swap pode ser de várias formas

1. Um arquivão de tamanho fixo:
   - Quando o arquivo enche, para tudo!
   - Mais fácil de otimizar que um arquivo comum, por ter tamanho fixo
   - Ex: Mach 2.0

Swap Space - 4

Swap pode ser de várias formas

2. Um arquivo de tamanho variável:
   - Quando o arquivo enche, é só crescê-lo
   - Mais difícil de otimizar
   - Ex: Mach 3.0

Swap Space - 5

Swap pode ser de várias formas

3. Uma partição separada:
   - Fácil de otimizar
   - Difícil de modificar
   - Ex: Linux

Conclusão

Memória virtual: permite tornar os processos independentes da quantidade de memoria disponível

Desempenho: bom se a quantidade de memória real disponível é adequada para evitar *misses*