BCC264

Sistemas Operacionais

Gerência de Memória

Prof. Charles Garrocho

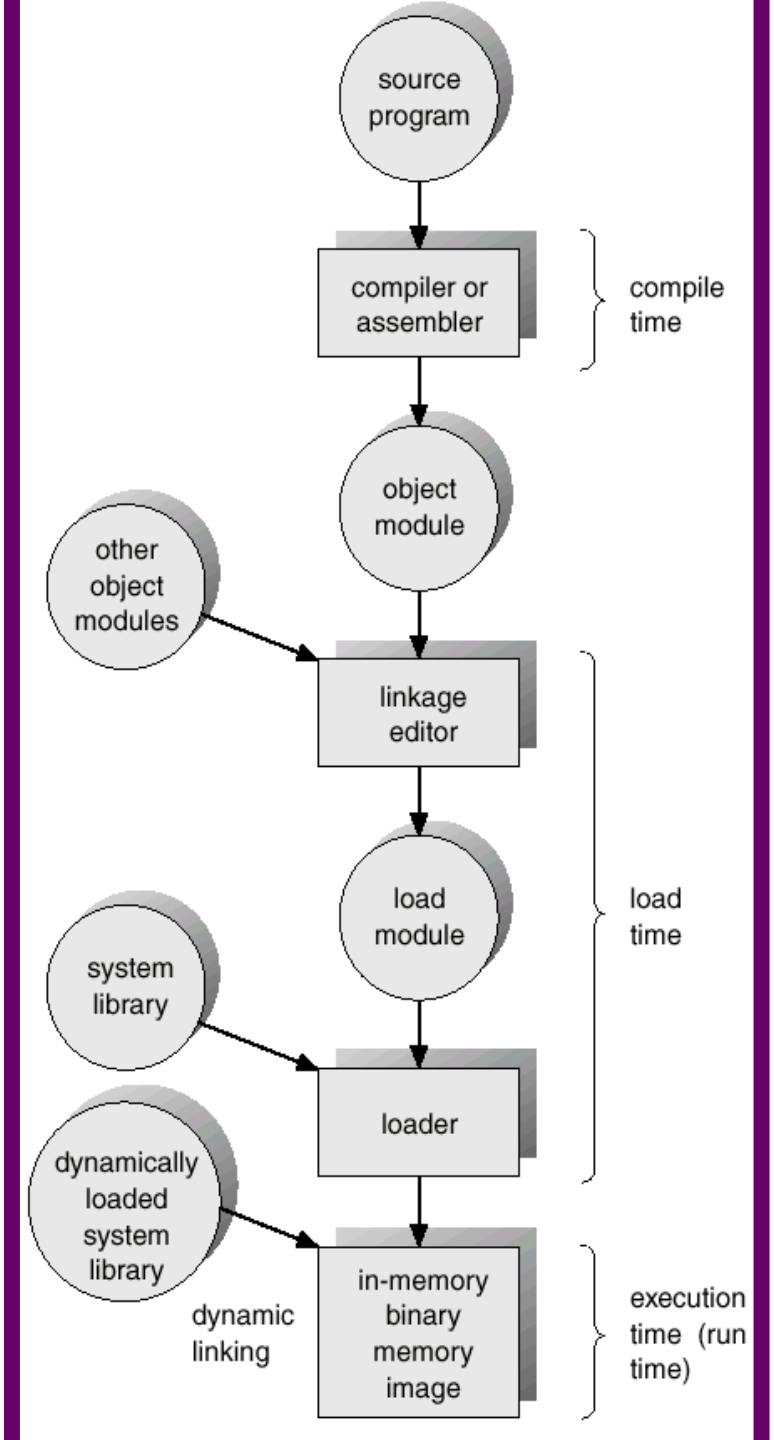
Conceitos básicos

- Programas precisam ser carregados na memória e associados a um processo para poderem ser executados
- Fila de entrada: processos que estejam no disco esperando para serem carregados na mem.
- Há vários passos a serem seguidos antes dos processos poderem executar

Vinculação de endereços (ligação)

- Programas em linguagem de alto nível (quase) nunca referenciam endereços específicos de memória:
  - C define variáveis globais, locais e dinâmicas, mas não define endereços específicos para elas
  - Dependendo da linguagem, do S.O. e do HW, há diversos momentos onde a associação entre os comandos e a memória pode acontecer

Passos no processamento de um programa até sua execução



Vinculação de endereços (ligação)

- Tempo de compilação
  - Se houver uma definição fixa sobre localização o compilador pode gerar endereços diretamente

- Problemas?
  - Se for preciso mudar os endereços, é preciso recompilar o programa

  - Comum para elementos com endereços definidos pelo HW (p.ex., vetor de interrupções)

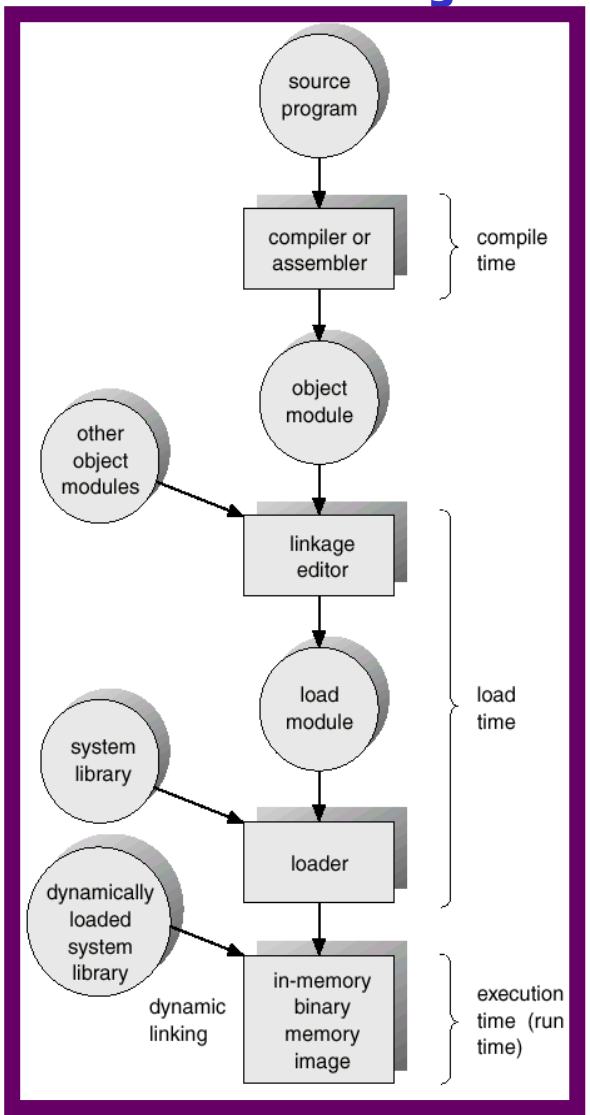
Vinculação de endereços (ligação)

- Tempo de carga:
  - Compilador gera anotações sobre acessos relacionados a certos endereços
  - O carregador (*loader*) lê essas anotações e altera o código do programa carregado para inserir os endereços adequados a cada caso
  - Endereços ganham valor a partir do momento que sabe-se a posição inicial do programa

Vinculação de endereços (ligação)

- Tempo de execução:
  - Em alguns casos a localização de um processo na memória pode mudar em tempo de execução
  - Nesse caso o HW deve prover suporte especial
  - É preciso refazer vínculos eficientemente
    - Mapas de endereços
    - Tabelas de relação
    - Endereçamento relativo

Passos no processamento de um programa até sua execução

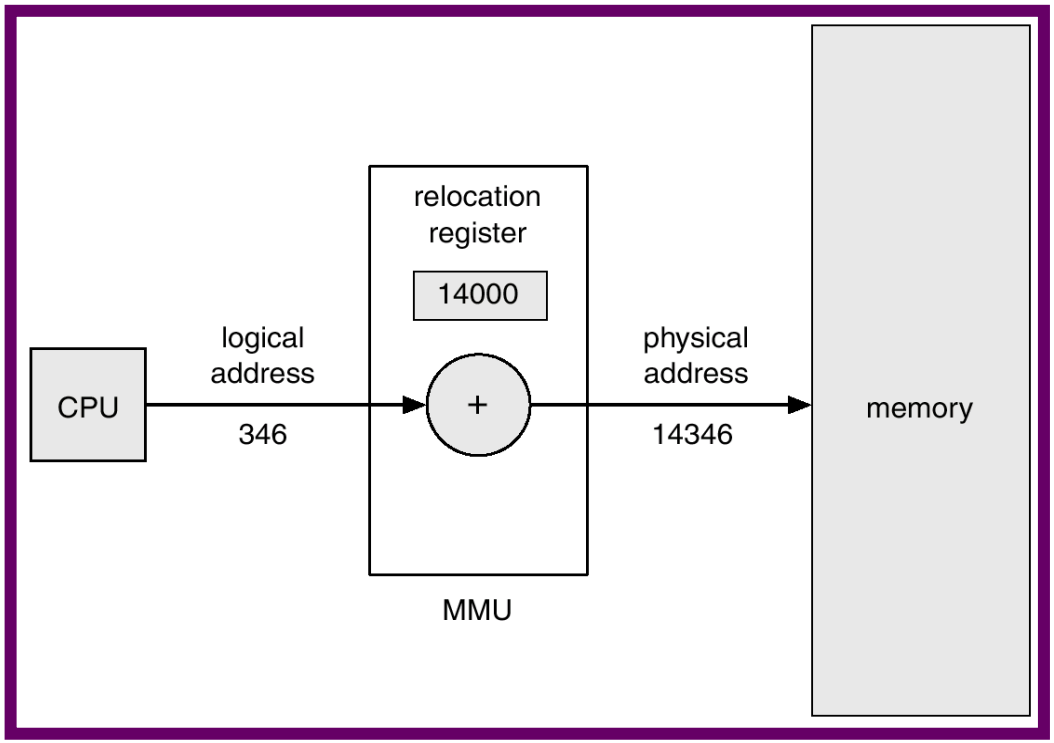


Endereços lógicos x físicos

- End. lógico: gerado pela CPU (virtual)
- End. físico: visto pela unidade de memória (real)
- Mapeamento depende de recursos do HW
  - Unidade de gerência de memória (MMU)
  - Em sistemas simples (sem MMU), virtual=real

Unidade de gerenciamento de memória (MMU)

- Endereços lógicos são transformados
  - Tabela de tradução
  - Registrador de relocação



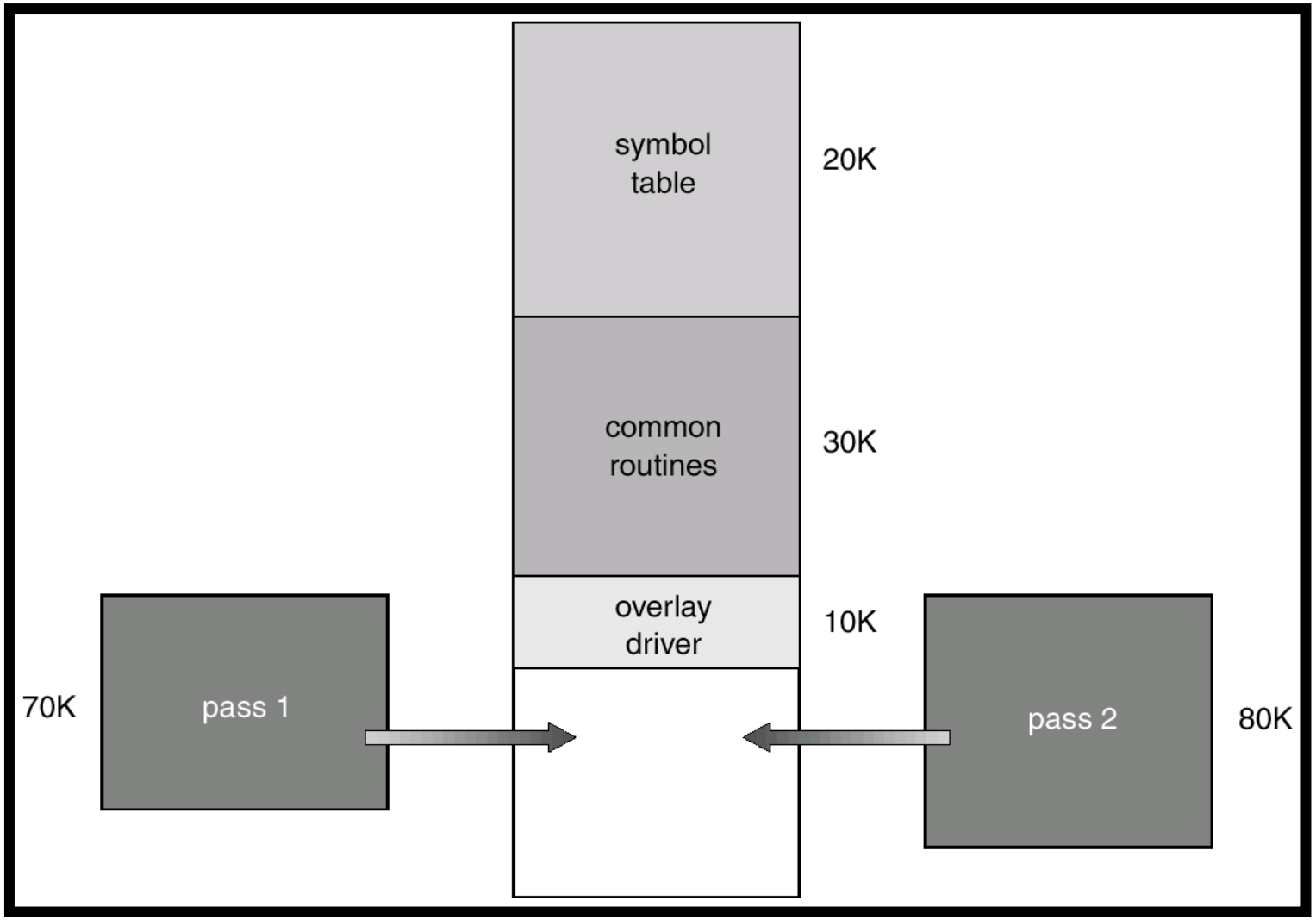
Carga dinâmica

- Permite que partes de um programa só sejam carregados na mem. quando chamadas
  - Melhor utilização da memória
  - Partes não utilizadas podem ser descarregadas

Sobreposição

- Mantém em memória apenas instruções e dados que são necessários em um dado momento
- Necessário quando processo é maior do que memória alocada a ele
- Pode ser implementado por usuário e não precisa de suporte de HW ou do S.O., mas o código pode ser complicado

Exemplo de sobreposição

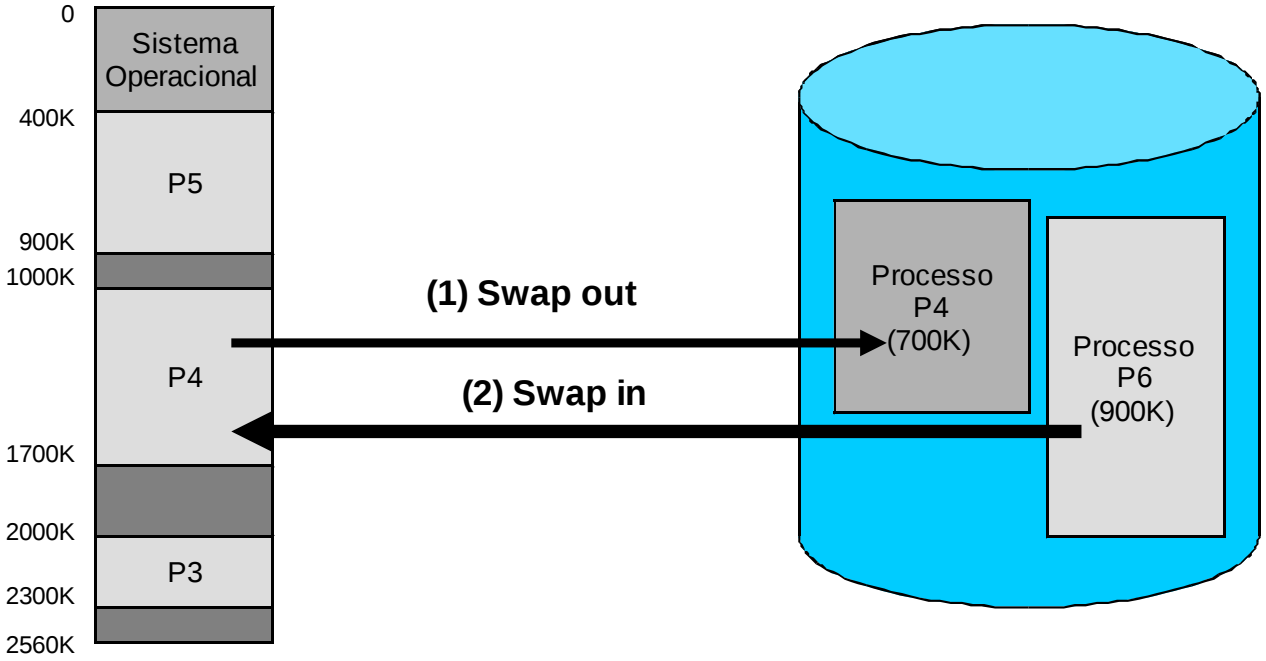


Swapping (permuta de processos)

- Processos inteiros podem ser transferidos temporariamente para memória secundária e posteriormente trazidos de volta
  - P.ex., quando são suspensos pelo escalonador
- Maior *overhead* é o tempo de transferência
- Presente em diversos S.O. (Unix, Linux, Win*...)

Swapping

- Processos suspensos fora da memória
- Exigem relocação ao serem recarregados

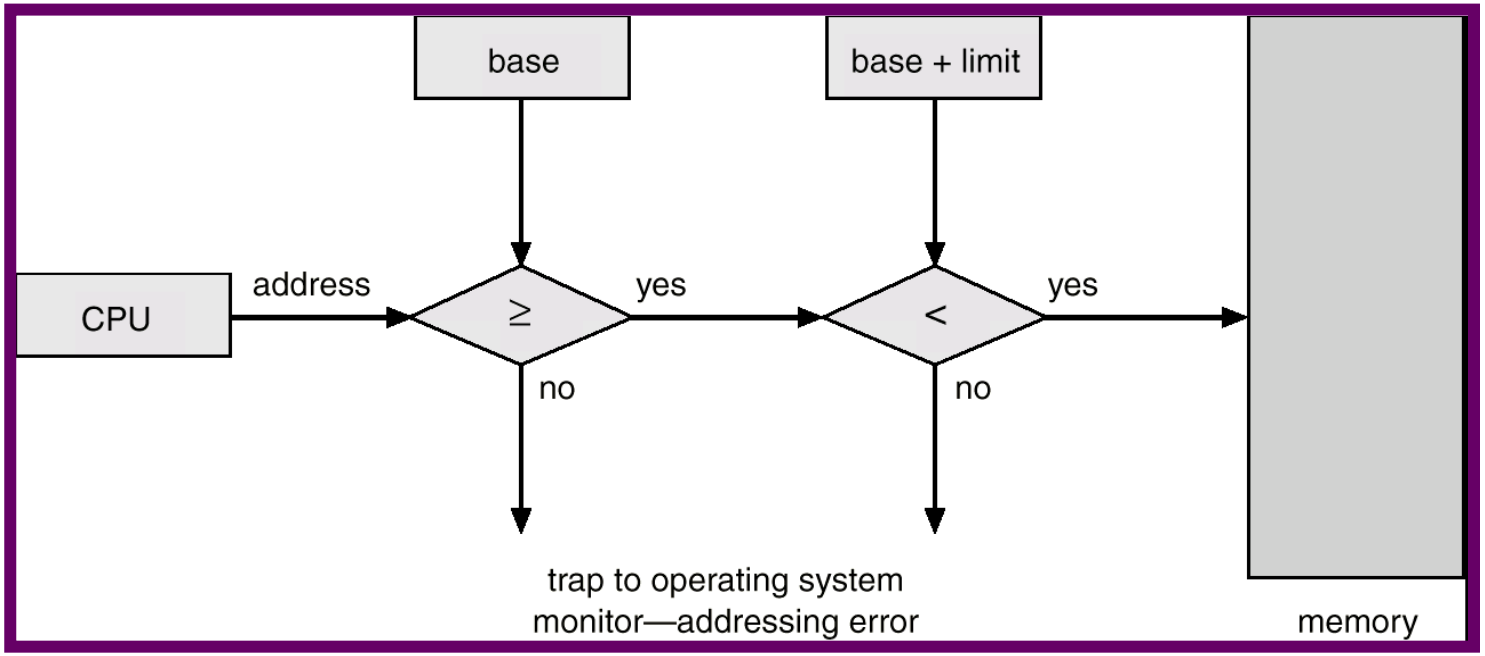


Alocação contígua com partição única

- Cada processo ocupa um bloco único da memória física
- Duas partições na memória
  - S.O. geralmente reside nas posições iniciais, onde fica vetor interrupções
  - Processo de usuário
- Tudo fica mais simples mas, para multiprogramação, é desejável que diversos programas estejam na memória

Proteção de memória

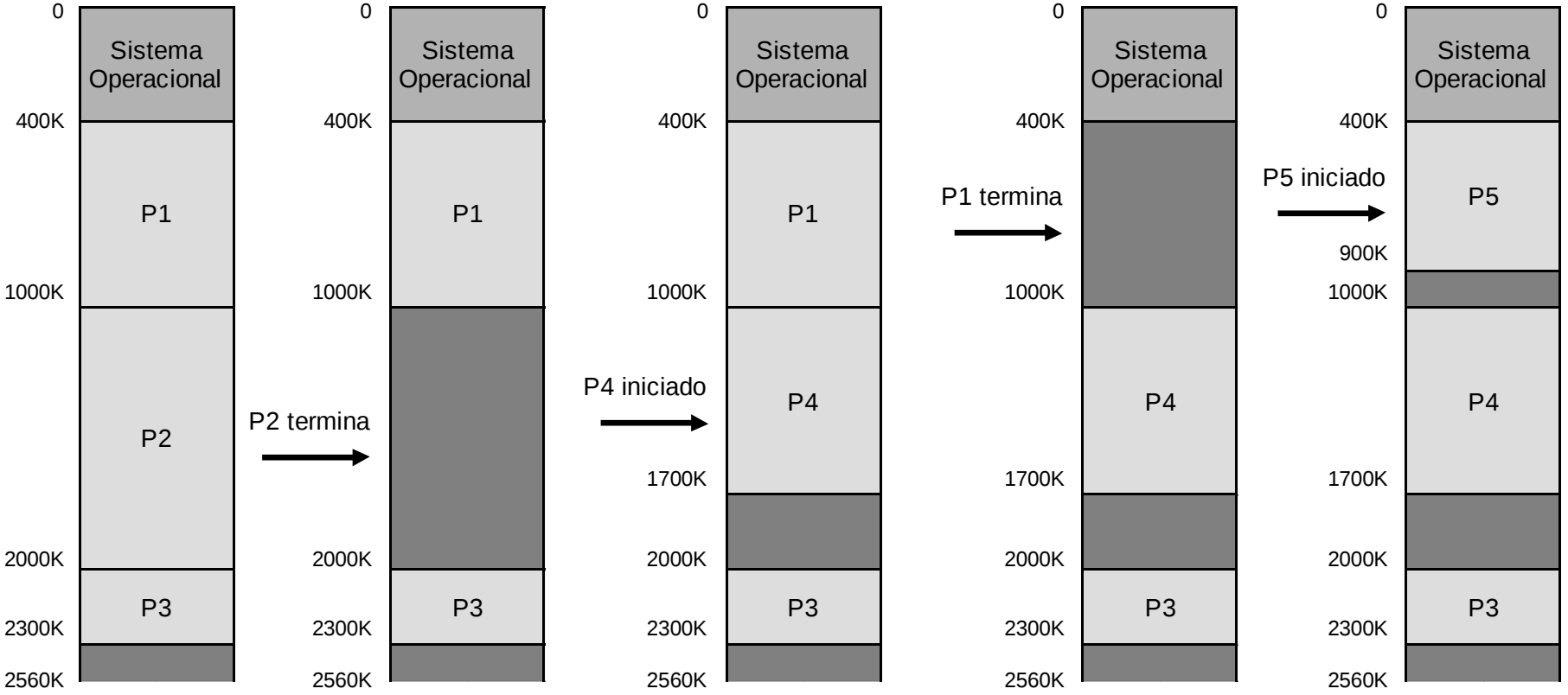
- Todo acesso é comparado à faixa definida pelos registradores base e limite
- Acessos fora da faixa geram TRAP



Alocação contígua com múltiplos processos

- Cada processo ocupa um bloco da memória
- Ao terminar, bloco é liberado (buraco)
- Processos que chegam ocupam buracos
- S.O. deve controlar partições e buracos
- Usada no OS/360

Alocação contígua com múltiplos processos



Problema de alocação dinâmica: onde colocar um processo novo?

- *First-fit* (primeiro apto): primeiro que couber
- *Best-fit* (mais apto): o de tamanho mais próximo
- *Worst-fit* (menos apto): sempre o maior buraco

First-fit e *best-fit* se saem melhor que *worst-fit* em termos de velocidade e utilização do espaço.

Fragmentação

- Quebra do espaço em frações não utilizáveis
- Fragmentação externa:
  - Memória não utilizada dividida em muitos buracos pequenos demais para serem úteis
  - Resultados empíricos e analíticos mostram 1/3 de desperdício de memória
- Fragmentação interna:
  - Custo de manipulação de bloco muito pequeno não compensa. Vários blocos pequenos torna-se um problema a mais.
    - Solução: aloca-se um pouco mais do que se precisa
  - Problema: buracos dentro do bloco

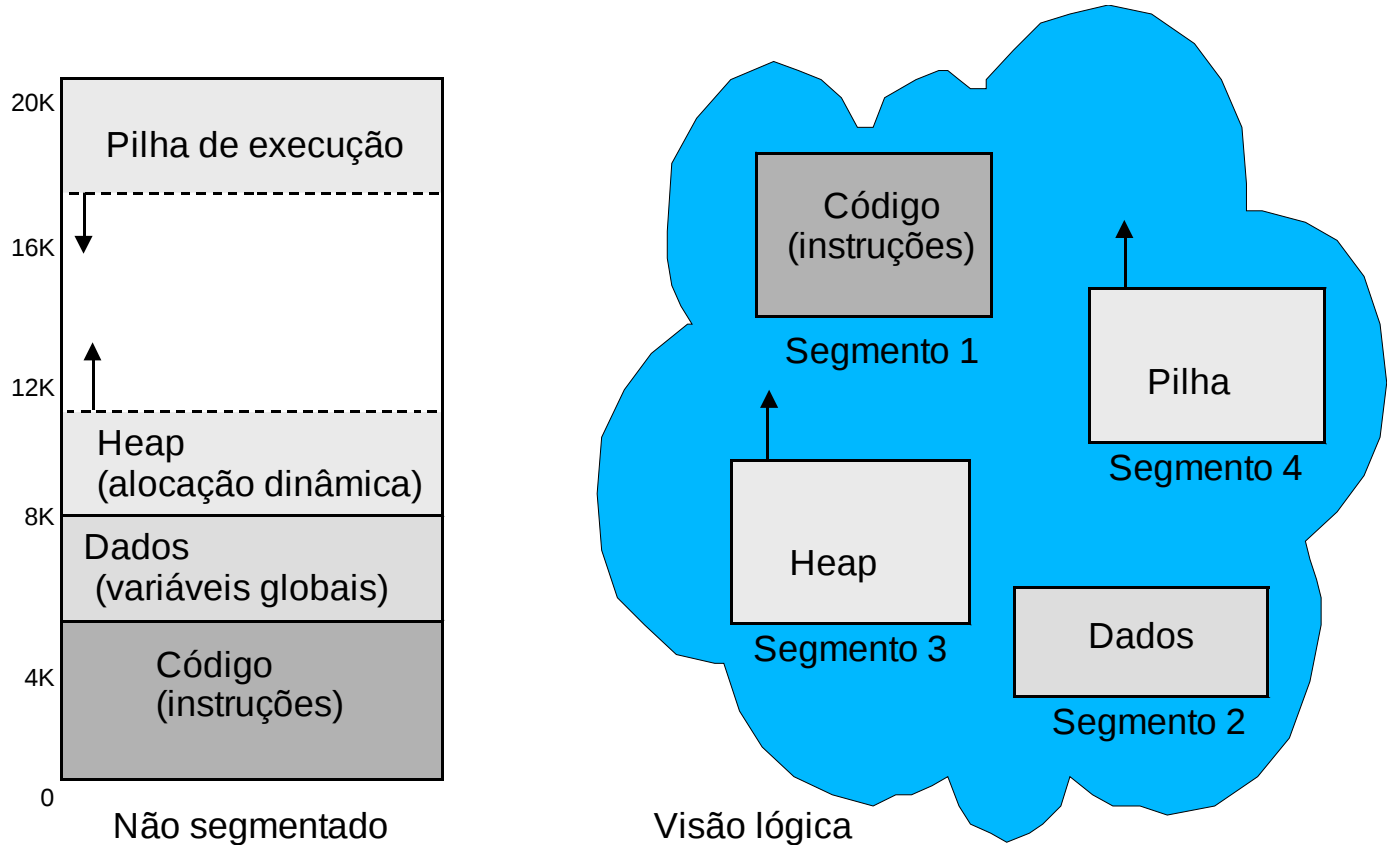
Compactação

- Solução para a fragmentação externa
  - Mover blocos ocupados para perto uns dos outros
  - Agrupar os buracos em um único bloco maior
  - Algoritmos de compactação devem tentar reduzir a quantidade de dados transferidos

Segmentação

- Divisão da memória do processo em unidades
- Baseado na visão lógica do usuário/programador
- Um programa é uma coleção de segmentos de memória independentes (código, dados, pilha, ...)
- Cada um pode ser acessado independentemente
  - Como se fossem diferentes dimensões
- Tornou-se “popular” ao ser adotada pela Intel

Segmentação

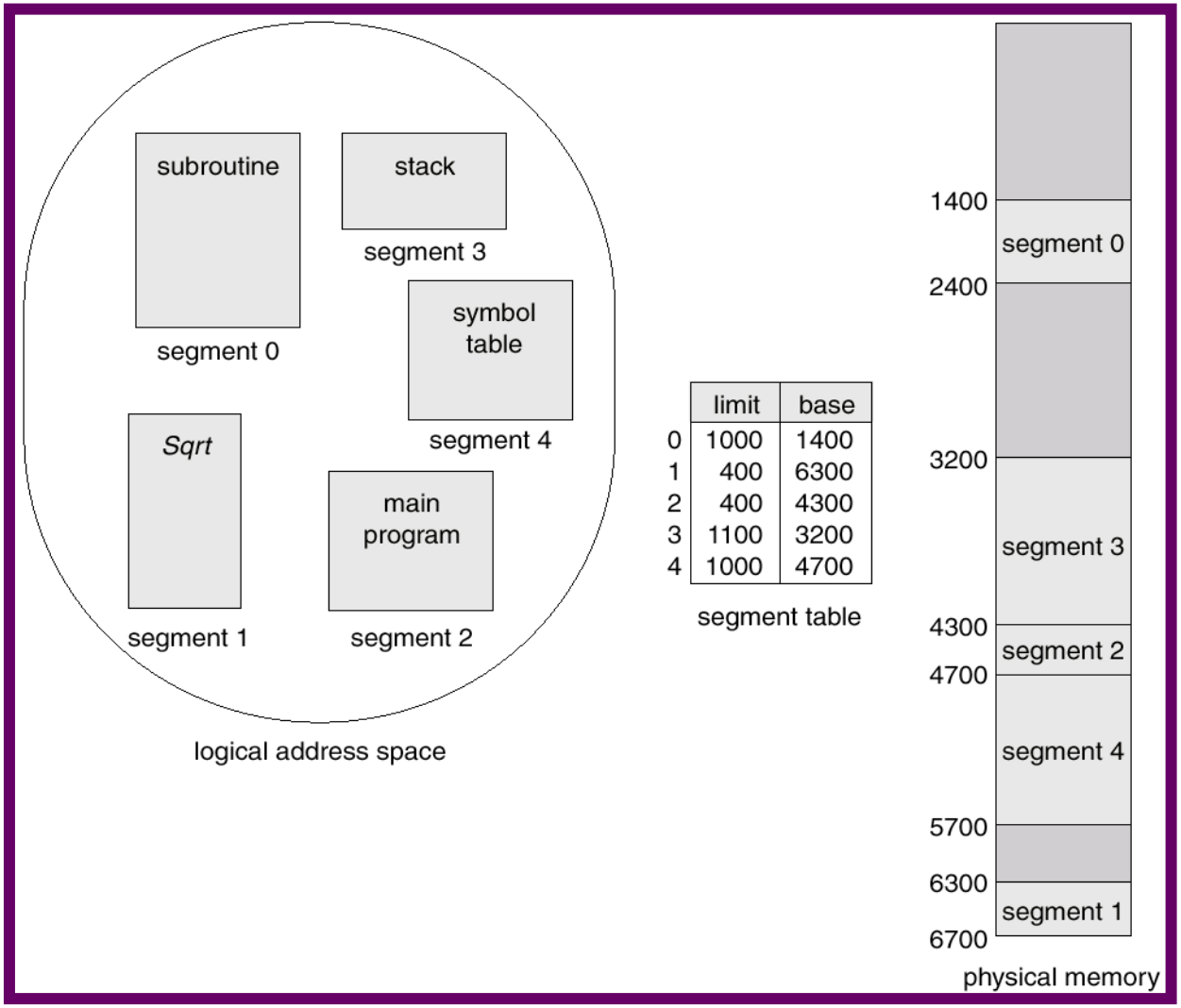


- Mesmos problemas de alocação contígua!

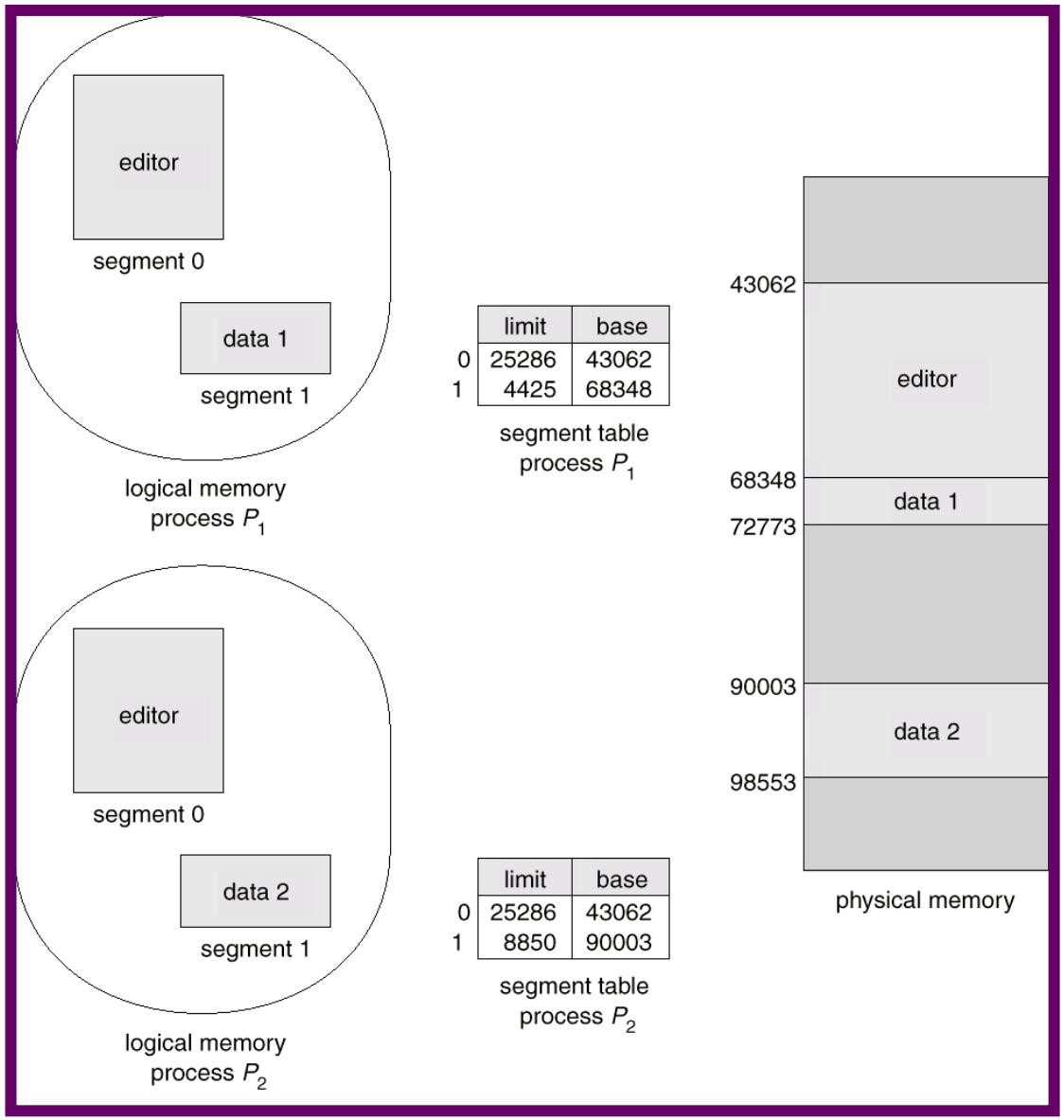
Arquitetura para segmentação

- Endereço lógico é uma dupla: <segm.,offset>
- Tabela de segmentos mapeia segmento em uma área da memória (base + limite)
- Segmentos: pré-definidos (fixos) ou enumeráveis
  - Enumeráveis: tabela de segmentos (com limite)
- Segmentos podem ser compartilhados

Segmentação: exemplo



Segmentos compartilhados

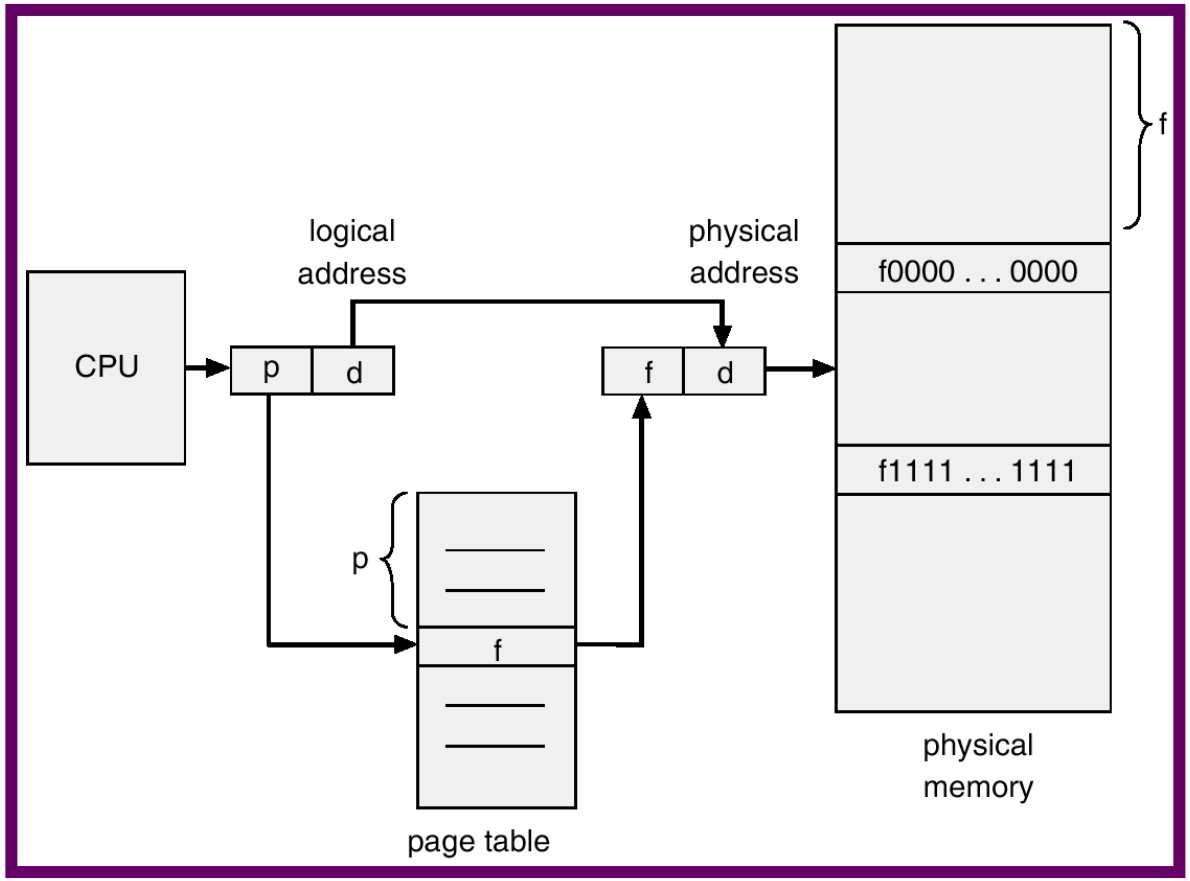


Paginação (*paging*)

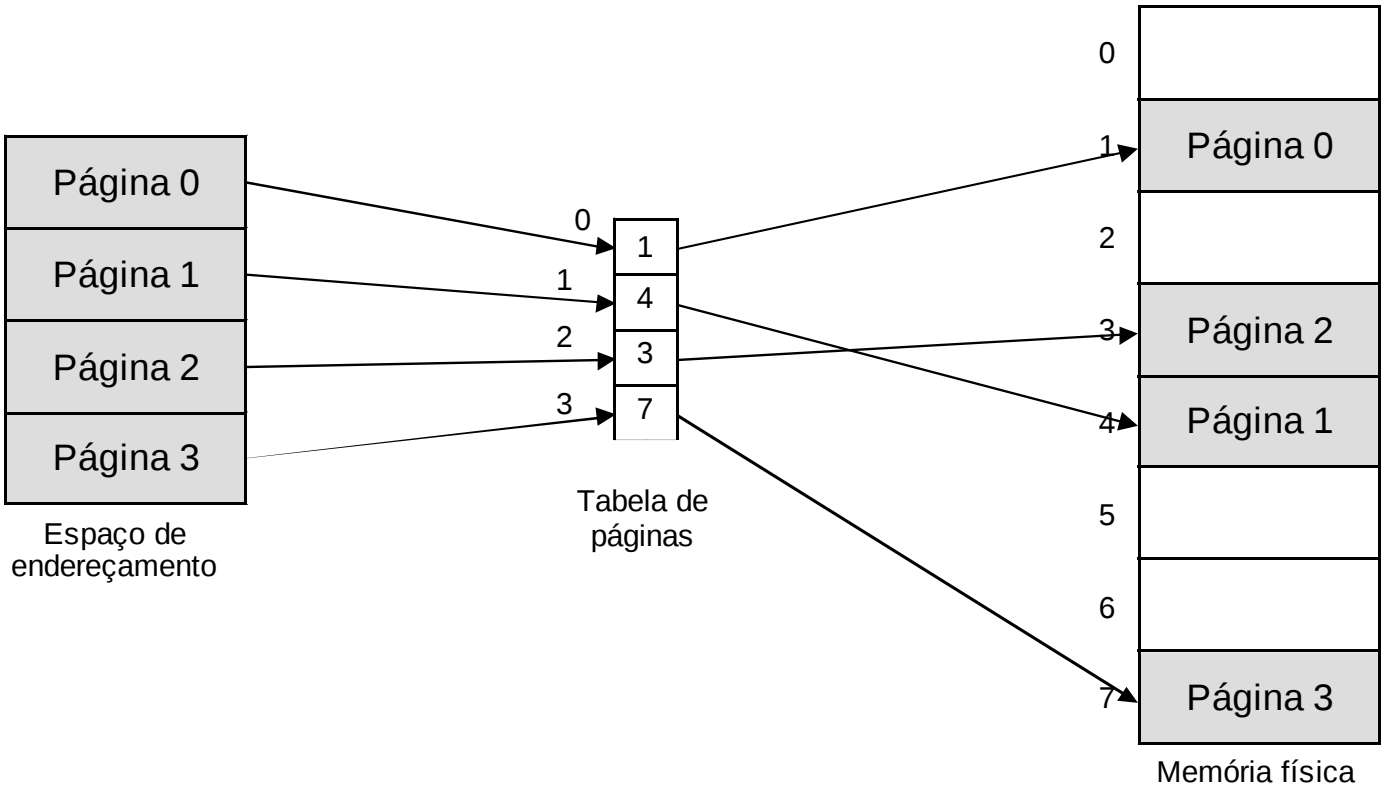
- Memória (lógica/física) é dividida em blocos de tamanho fixo – p.ex., 4 KB
- Blocos lógicos (páginas) são mapeadas em blocos físicos (quadros) pelo HW
- Endereços lógicos contíguos podem estar em páginas diferentes, em quadros não contíguos
- Quadros vazios são gerenciados

Arquitetura de tradução de endereços

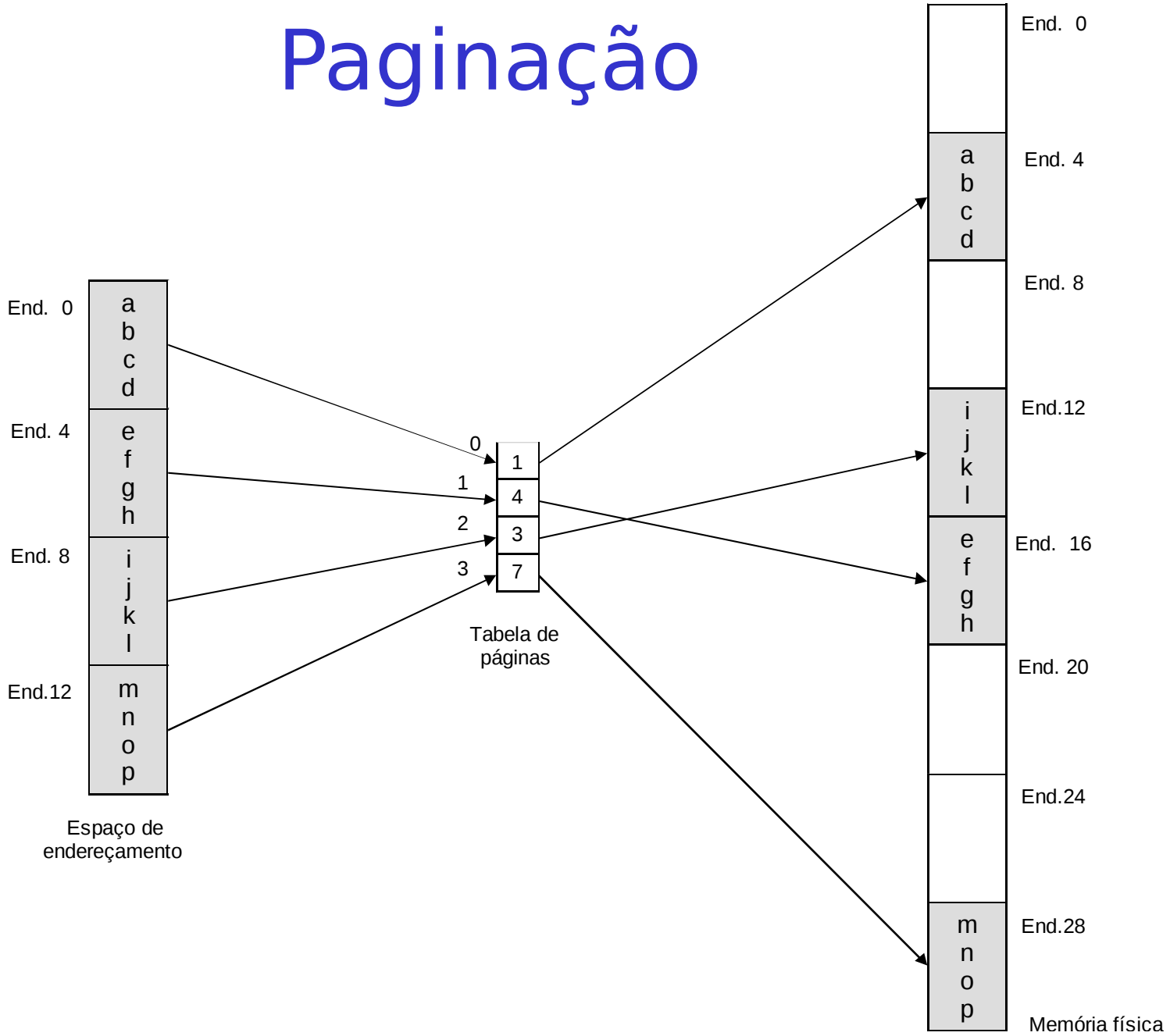
- Endereço lógico é dividido em <página,deslocamento>



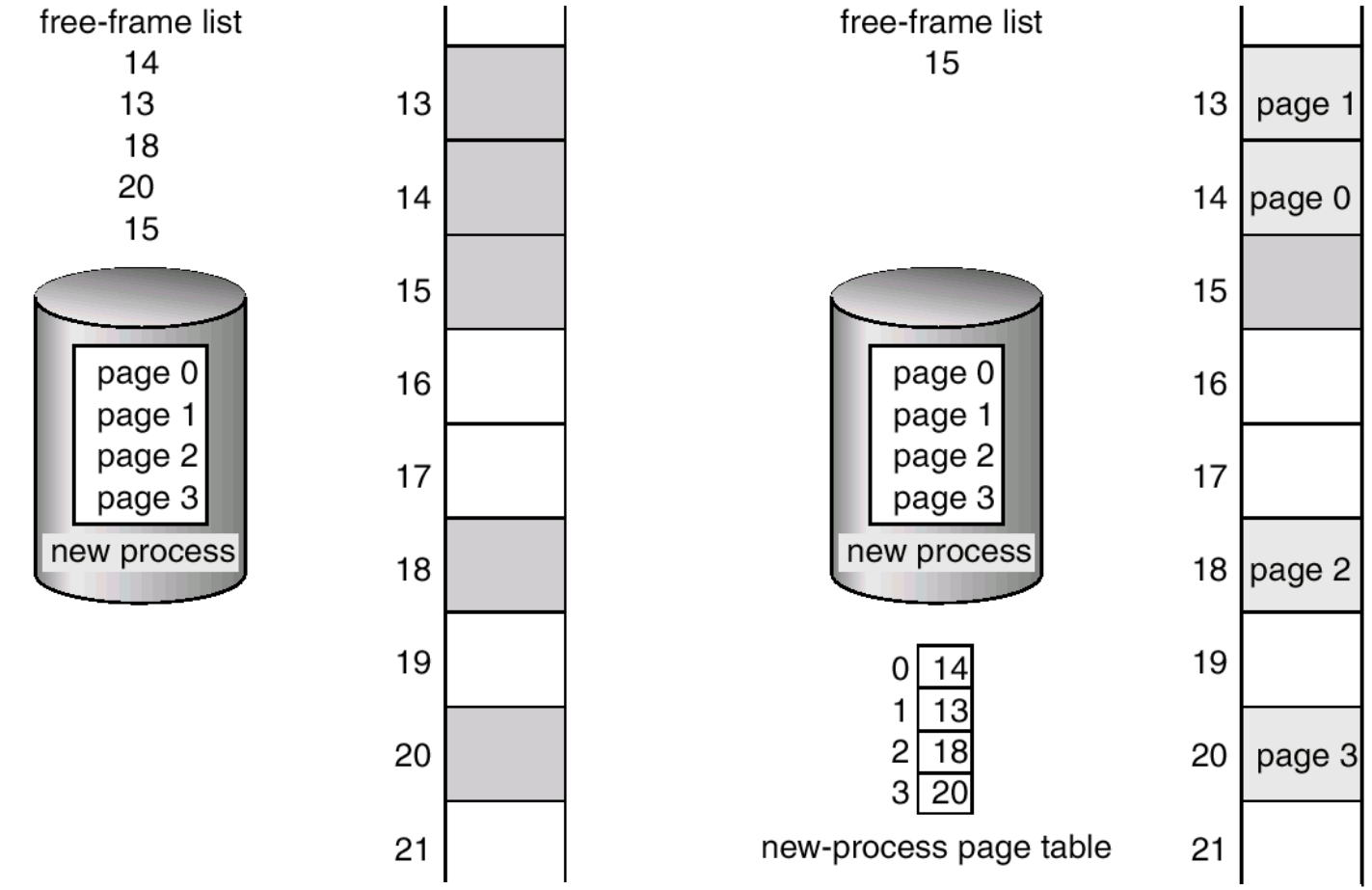
Paginação


Página 0
Página 1
Página 2
Página 3
Espaço de endereçamento
0
1
2
3
1
4
3
7
Tabela de páginas
0
1
2
3
4
5
6
7
Página 0
Página 2
Página 1
Página 3
Memória física

Paginação


End. 0
a
b
c
d
End. 4
e
f
g
h
End. 8
i
j
k
l
End.12
m
n
o
p
Espaço de endereçamento
0
1
1
4
2
3
3
7
Tabela de páginas
End. 0
a
b
c
d
End. 4
End. 8
i
j
k
l
End.12
e
f
g
h
End. 16
End. 20
End.24
m
n
o
p
End.28
Memória física

Gerência de quadros livres



Antes da alocação

Depois da alocação

Compartilhamento de páginas

- 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

Compartilhamento de páginas

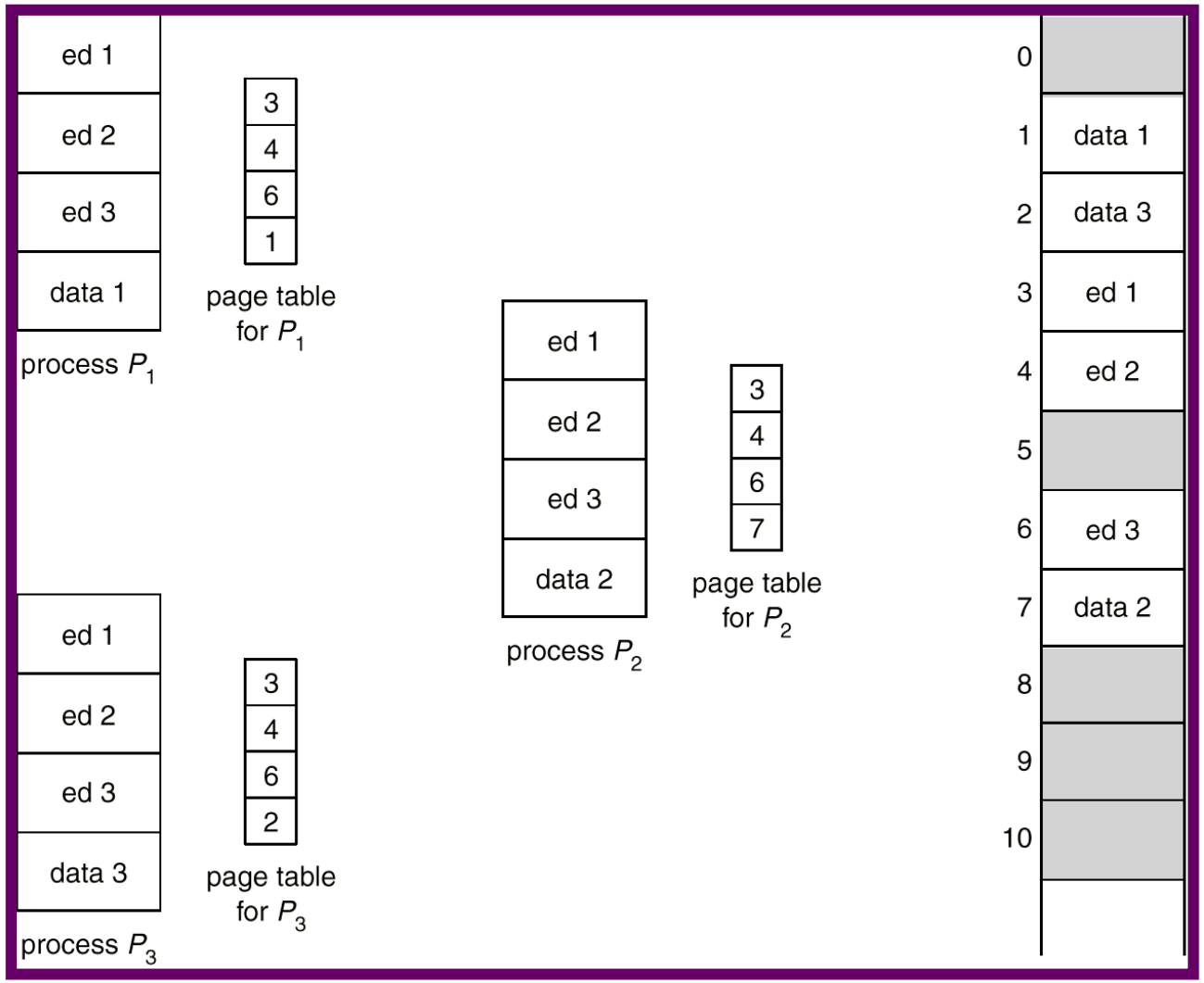


Tabela de páginas

- Tabela fica na memória
- Registrador especial aponta para ela (PTBR)
- Dois acessos à mem. para cada leitura/escrita
  - Um para acessar a tabela
  - Outro para acessar a página
- Hardware especial (cache) evita 1o. acesso
  - *Translation Look-aside Buffer (TLB)*

Hardware de paginação com TLB