Capítulo 8: Memória principal

Capítulo 8: Gerenciamento de memória

- Base
- Swapping
- Alocação de memória contígua
- Paginação
- Estrutura da tabela de página
- Segmentação

Objetivos

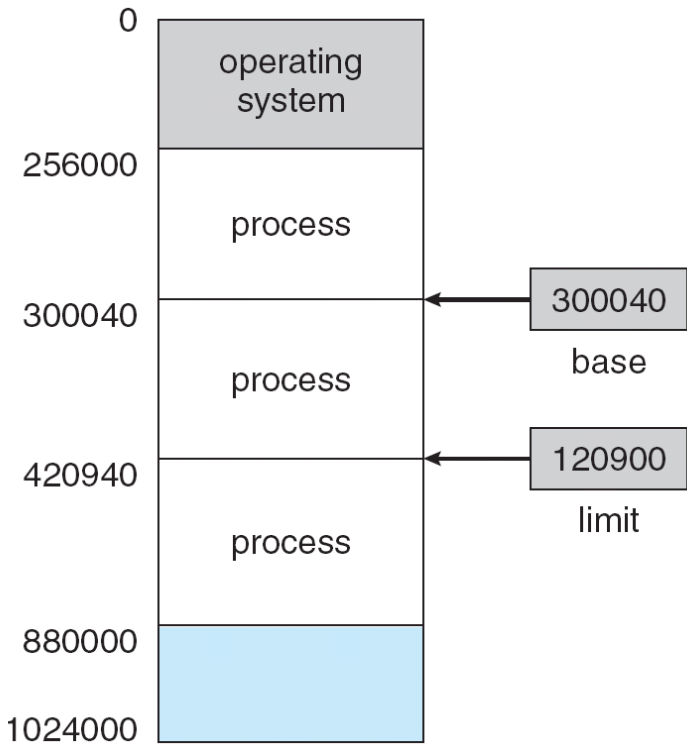
- Fornecer uma descrição detalhada de várias maneiras de organizar o hardware de memória.
- Discutir diversas técnicas de gerência de memória, incluindo paginação e segmentação.

Fatos básicos

- O programa deve ser carregado do disco para a memória e colocado dentro de um processo, para ser executado
- A memória principal e os registradores são o único armazenamento que a CPU pode acessar diretamente
- Acesso ao registrador em um clock de CPU (ou menos)
- A memória principal pode tomar muitos ciclos
- **Cache** fica entre a memória principal e os registradores da CPU
- Proteção da memória exigida para garantir a operação correta

Registradores de base e limite

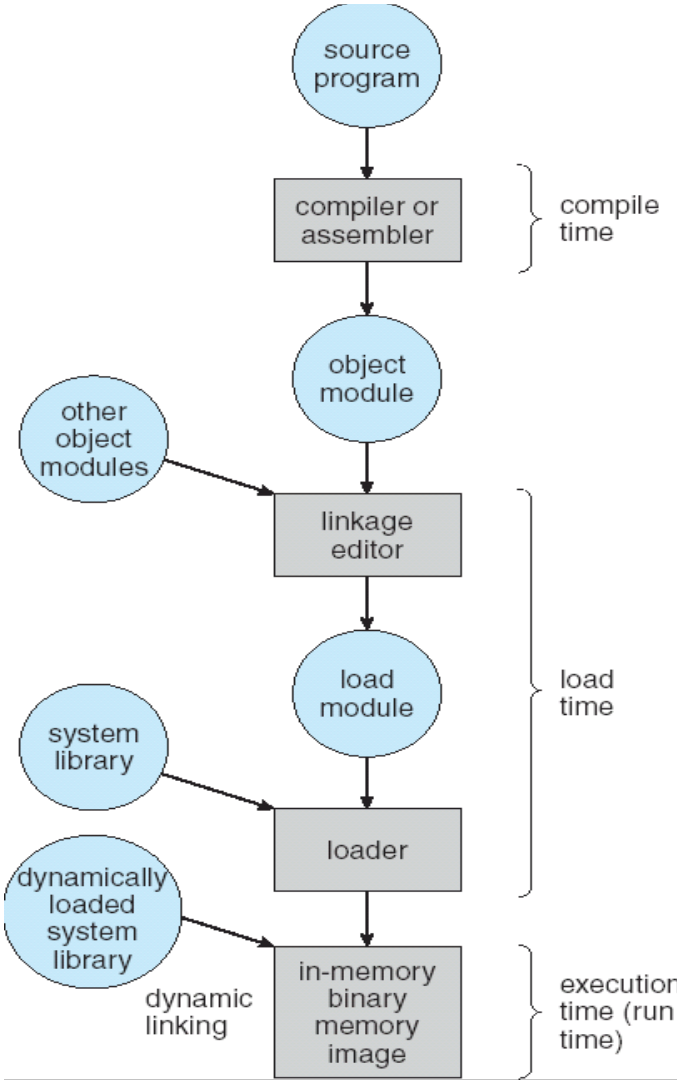
- Um par de registradores de **base** e **limite** definem o espaço de endereços lógicos



Associação de instruções e dados à memória

- A associação de endereços de instruções e dados a endereços de memória pode acontecer em três estágios diferentes:
  - **Tempo de compilação**: Se o local da memória for conhecido *a priori*, o **código absoluto** pode ser gerado; código deve ser recompilado se o local inicial mudar
  - **Tempo de carga**: Deve gerar **código relocável** se o local da memória não for conhecido durante a compilação
  - **Tempo de execução**: Associação adiada até a execução se o processo puder ser movido durante sua execução de um segmento da memória para outro. Precisa de suporte do hardware para mapas de endereço (por exemplo, registradores de base e limite)

Processamento multi-etapas de um programa



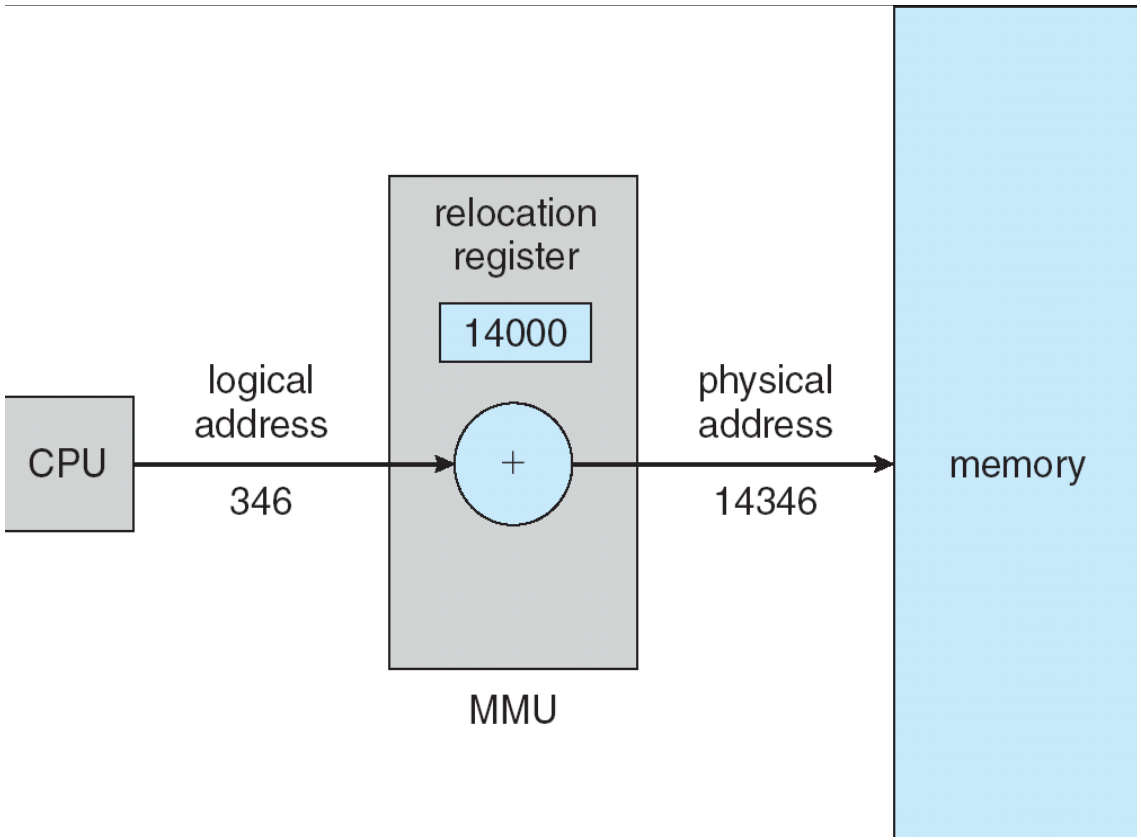
Espaço de endereços lógicos e físicos

- O conceito de um espaço de endereço lógico vinculado a um **espaço de endereço físico** separado é central ao gerenciamento de memória apropriado
  - **Endereço lógico** – gerado pela CPU; também conhecido como **endereço virtual**
  - **Endereço físico** – endereço visto pela unidade de memória

Unidade de gerenciamento de memória (MMU)

- Dispositivo de hardware que mapeia endereço virtual para físico.
- No esquema MMU, o valor no registrador de relocação é somado a cada endereço gerado por um processo do usuário no momento em que é enviado à memória
- O programa do usuário lida com endereços *lógicos*; ele nunca vê os endereços físicos *reais*

Relocação dinâmica usando MMU



Carregamento dinâmico

- Rotina não é carregada até ser chamada
- Melhor utilização de espaço da memória (rotina não usada nunca é carregada)
- Útil quando grandes quantidades de código são necessárias para lidar com casos que ocorrem com pouca freqüência
- Nenhum suporte especial do sistema operacional precisa ser implementado. A responsabilidade do carregamento é do programa do usuário.

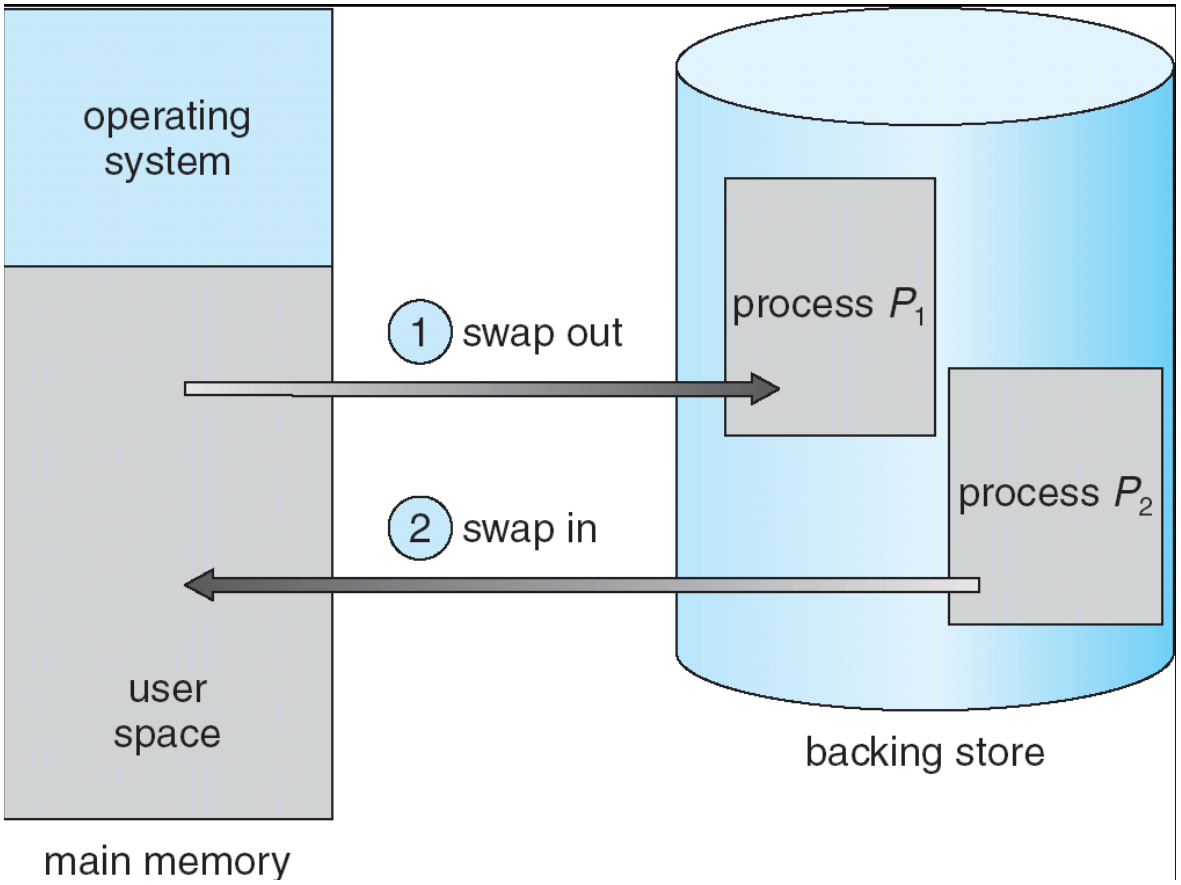
Vínculo dinâmico (bibliotecas compartilhadas)

- Vínculo adiado até o tempo da execução
- Pequeno pedaço de código, *stub*, usado para localizar a rotina de biblioteca apropriada residente na memória
- Stub substituído pelo endereço da rotina, e executa a rotina
- SO necessário para verificar se a rotina está no endereço de memória dos processos
- Vínculo dinâmico é particularmente útil para bibliotecas (multiplas versões, uma única instância por versão).
- Sistema também conhecido como **bibliotecas compartilhadas**

Swapping

- Um processo pode ser trocado temporariamente entre a memória e um armazenamento de apoio, e depois trazido de volta para a memória para continuar a execução
- **Armazenamento de apoio** – disco rápido, grande o suficiente para acomodar cópias de todas as imagens da memória para todos os usuários; deve oferecer acesso direito a essas imagens da memória
- A parte principal do tempo de swap é o tempo de transferência; o tempo de transferência total é diretamente proporcional à quantidade de memória trocada

Visão esquemática do swapping

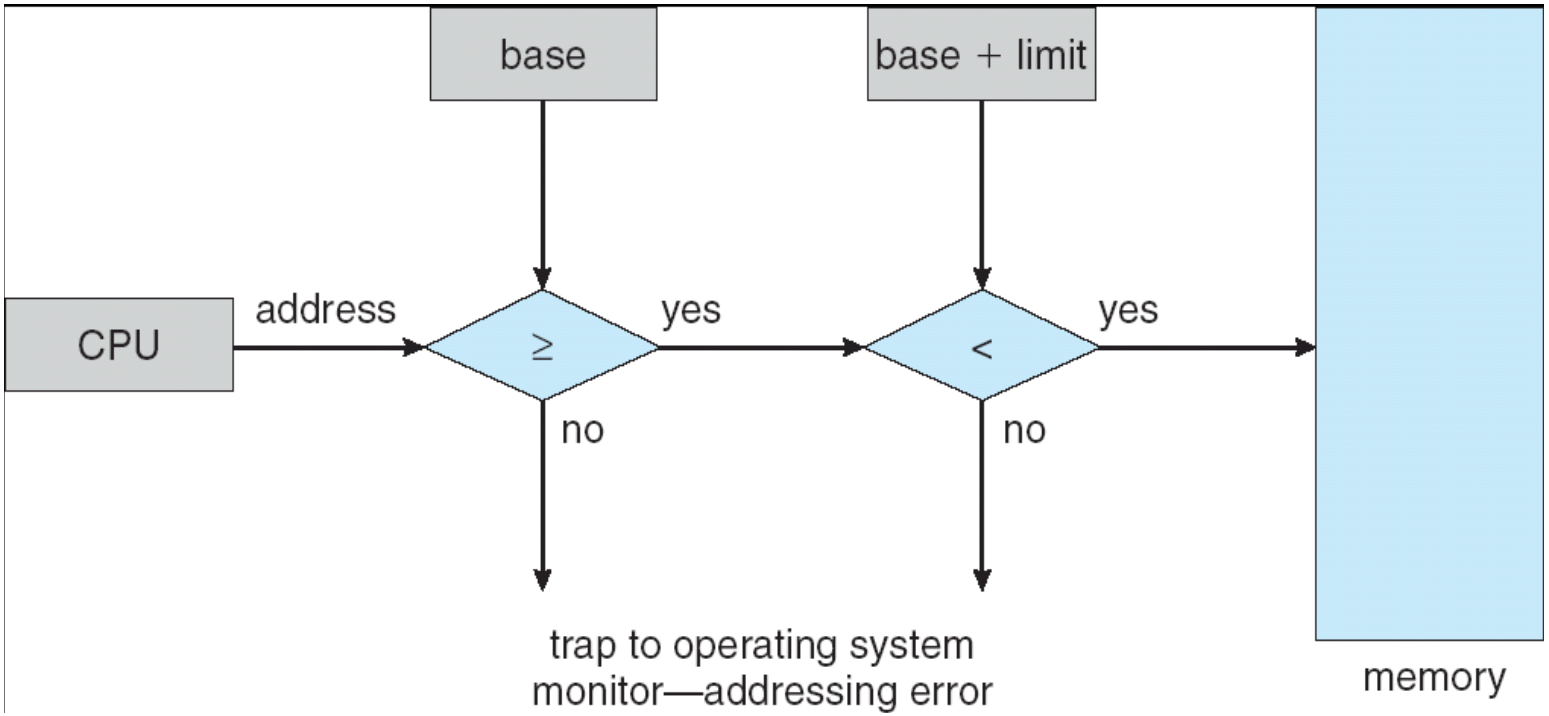


Alocação contígua

- Memória principal normalmente em duas partições:
  - SO normalmente mantido na memória baixa
  - Processos do usuário mantidos então na memória alta

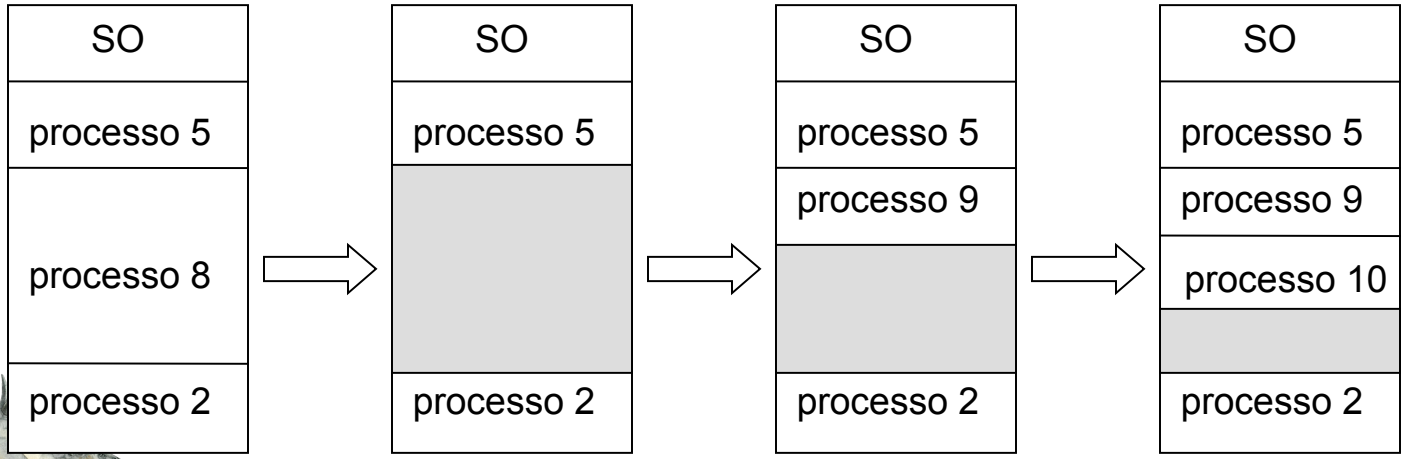
- Registradores de relocação usados para proteger acessos do usuário um do outro, e de alterarem o código e dados do SO
  - Registrador de base contém valor do menor endereço físico
  - Registrador de limite contém intervalo de endereços lógicos – cada endereço lógico precisa ser menor que o registrador de limite
  - MMU mapeia endereço lógico *dinamicamente*

Proteção da memória com registradores de base e limite



Alocação contígua (cont.)

- Alocação de múltiplas partições
  - Buraco – bloco de memória disponível; buracos de vários tamanhos estão espalhados pela memória
  - Quando um processo chega, ele recebe memória de um buraco grande o suficiente para acomodá-lo
  - O sistema operacional mantém informações sobre:
    a) partições alocadas b) partições livres (buracos)



Alocação de armazenamento dinâmico

Como satisfazer uma requisição de tamanho *n* de uma lista de buracos livres?

- **First-fit**: Aloca o *primeiro* buraco com tamanho suficiente
- **Best-fit**: Aloca o *menor* buraco com tamanho suficiente; deve procurar lista inteira, a menos que ordenado por tamanho
  - Produz o menor buraco restante
- **Worst-fit**: Aloca o *maior* buraco; também deve pesquisar lista inteira
  - Produz o maior buraco restante

Fragmentação

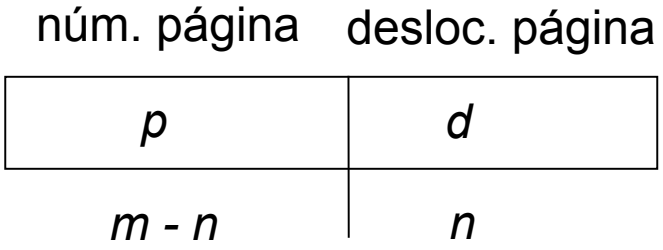
- **Fragmentação externa** – existe espaço de memória total para satisfazer uma solicitação, mas não é contíguo
- **Fragmentação interna** – memória alocada pode ser ligeiramente maior que a memória requisitada; essa diferença de tamanho é memória interna a uma partição, mas não está sendo usada
- Reduza a fragmentação externa com a **compactação**
  - Misture o conteúdo da memória para colocar toda a memória livre junta em um bloco grande
  - A compactação *só* é possível se a relocação for dinâmica, e é feita em tempo de execução

Paginação

- Espaço de endereçamento lógico de um processo pode ser não contíguo; processo recebe memória física sempre que houver memória disponível
- Divida a memória física em blocos de tamanho fixo, denominados **quadros** (tamanho é potência de 2, entre 512 bytes e 8.192 bytes)
- Divida a memória lógica em blocos do mesmo tamanho, denominados **páginas**
- Acompanhe todos os quadros livres
- Para executar um programa com tamanho de ***n*** páginas, precisa encontrar *n* quadros livres e carregar o programa
- Configure uma tabela de página para traduzir endereços lógicos para físicos
- Problema: fragmentação interna (se não usa a página toda)

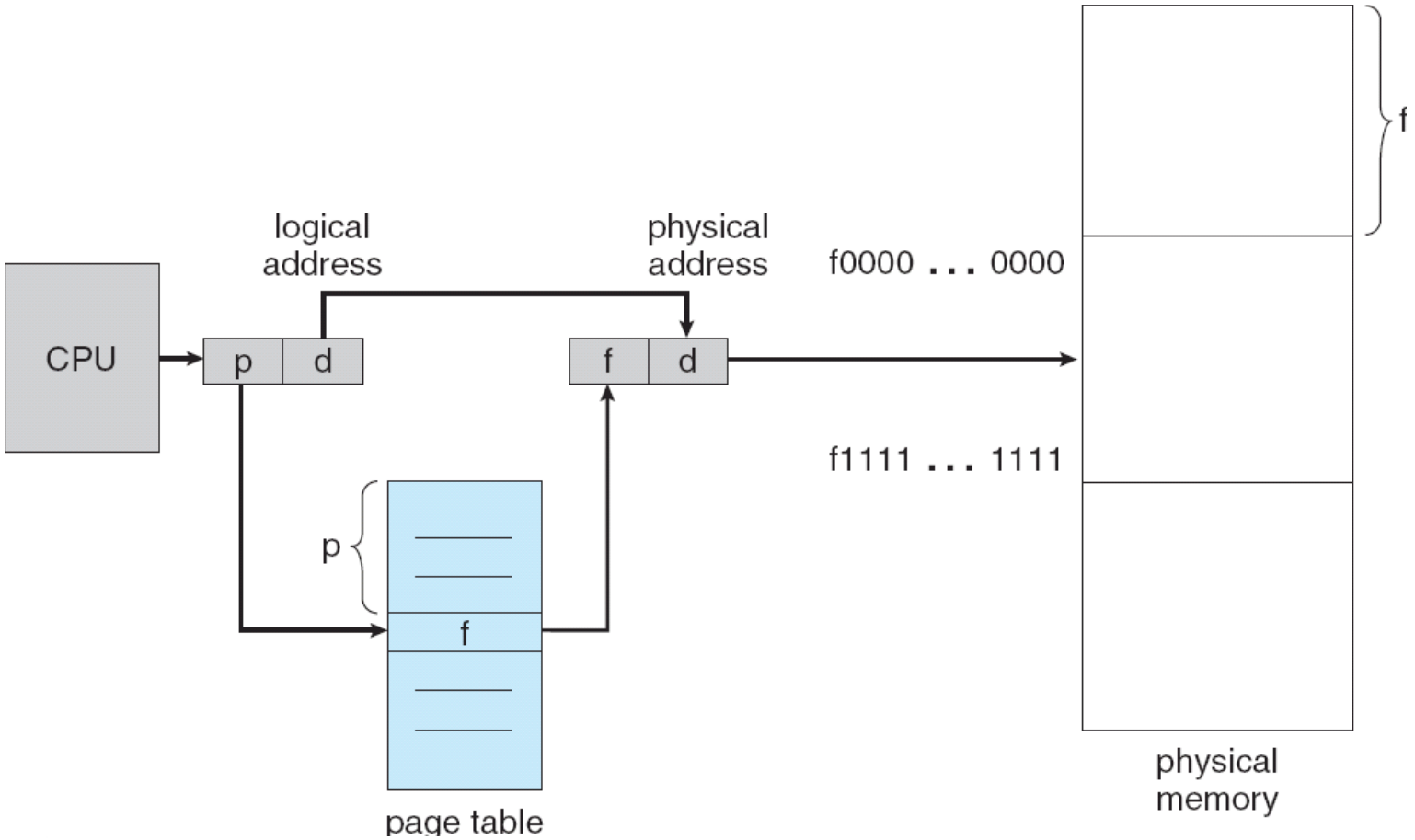
Esquema de tradução de endereço

- O endereço gerado pela CPU é dividido em:
  - **Número de página (*p*)** – usado como um índice para uma *tabela de página* que contém endereço de base de cada página na memória física
  - **Deslocamento de página (d)** – combinado com endereço de base para definir o endereço de memória físico que é enviado à unidade de memória

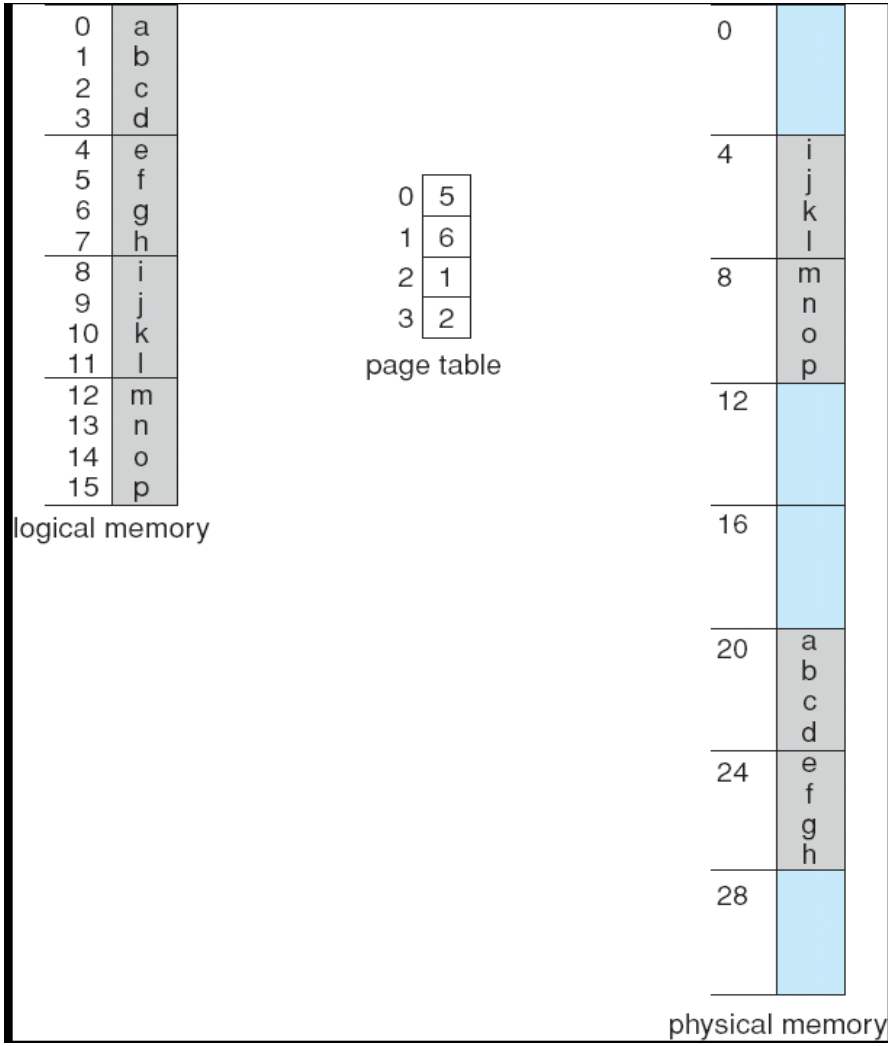


  - Para determinado espaço de endereçamento lógico 2^m e tamanho de página 2^n

Hardware de paginação

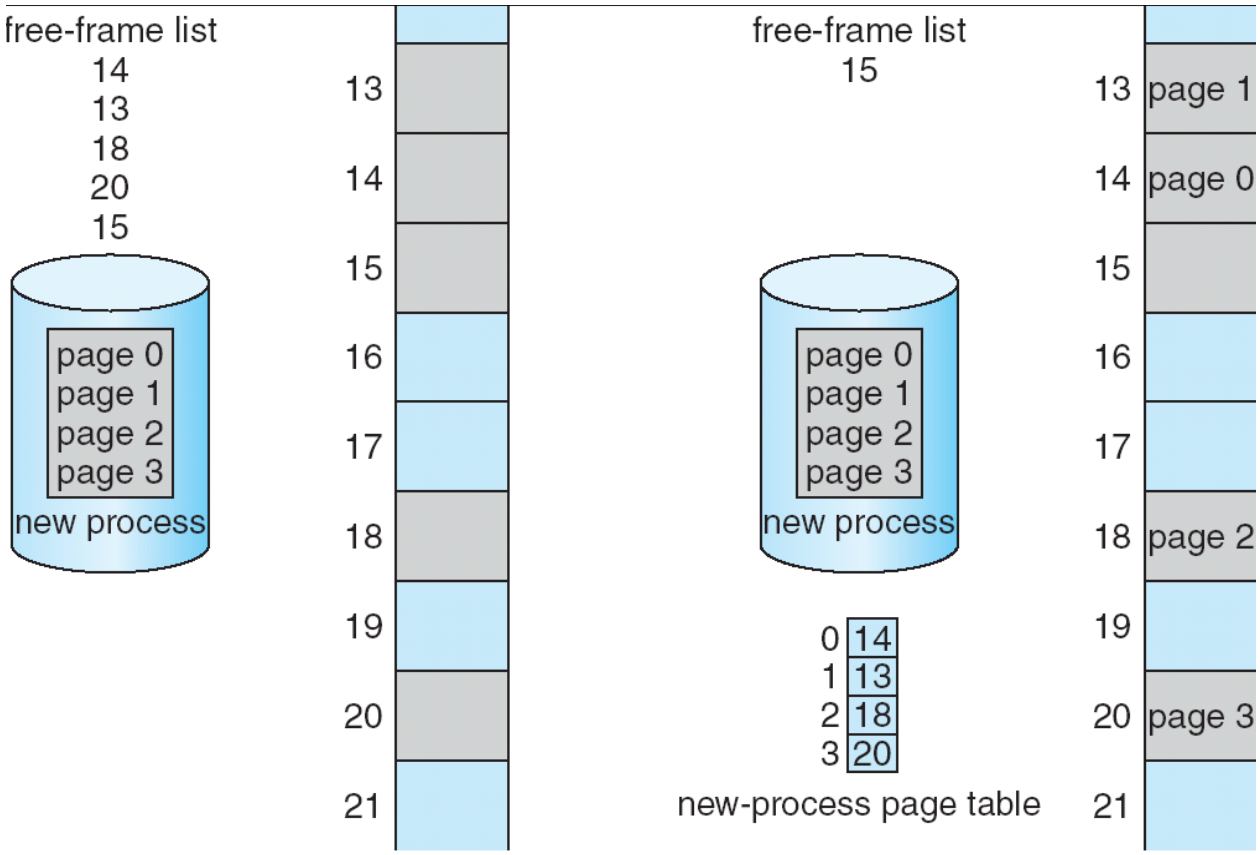


Exemplo de paginação



memória de 32 bytes e páginas de 4 bytes

Quadros livres

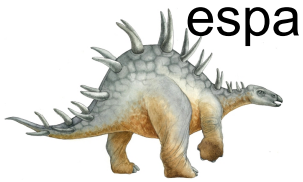


Antes da alocação

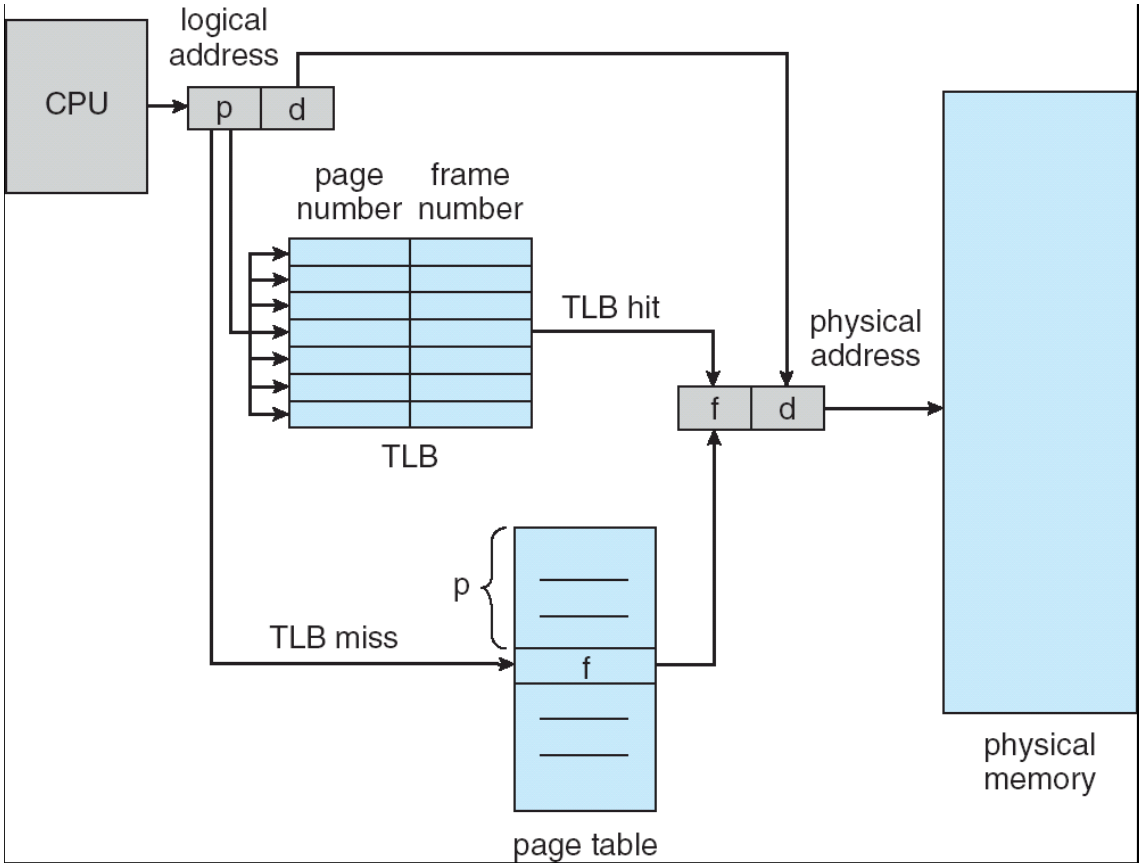
Após a alocação

Implementação da tabela de página

- A tabela de página é mantida na memória principal
- **Registrador de base da tabela de página (PTBR)** aponta para a tabela de página
- **Registrador de tamanho da tabela de página (PRLR)** indica tamanho da tabela de página
- Nesse esquema, cada acesso de dado/instrução exige dois acessos à memória: um para a tabela de página e um para o dado/instrução.
- O problema dos dois acessos à memória pode ser solucionado pelo uso de um cache de hardware especial para pesquisa rápida, chamado **memória associativa** ou **translation look-aside buffers (TLBs)**
- Alguns TLBs armazenam **identificadores de espaço de endereço (ASIDs)** em cada entrada de TLB – identifica exclusivamente cada processo para fornecer proteção do espaço de endereço para esse processo



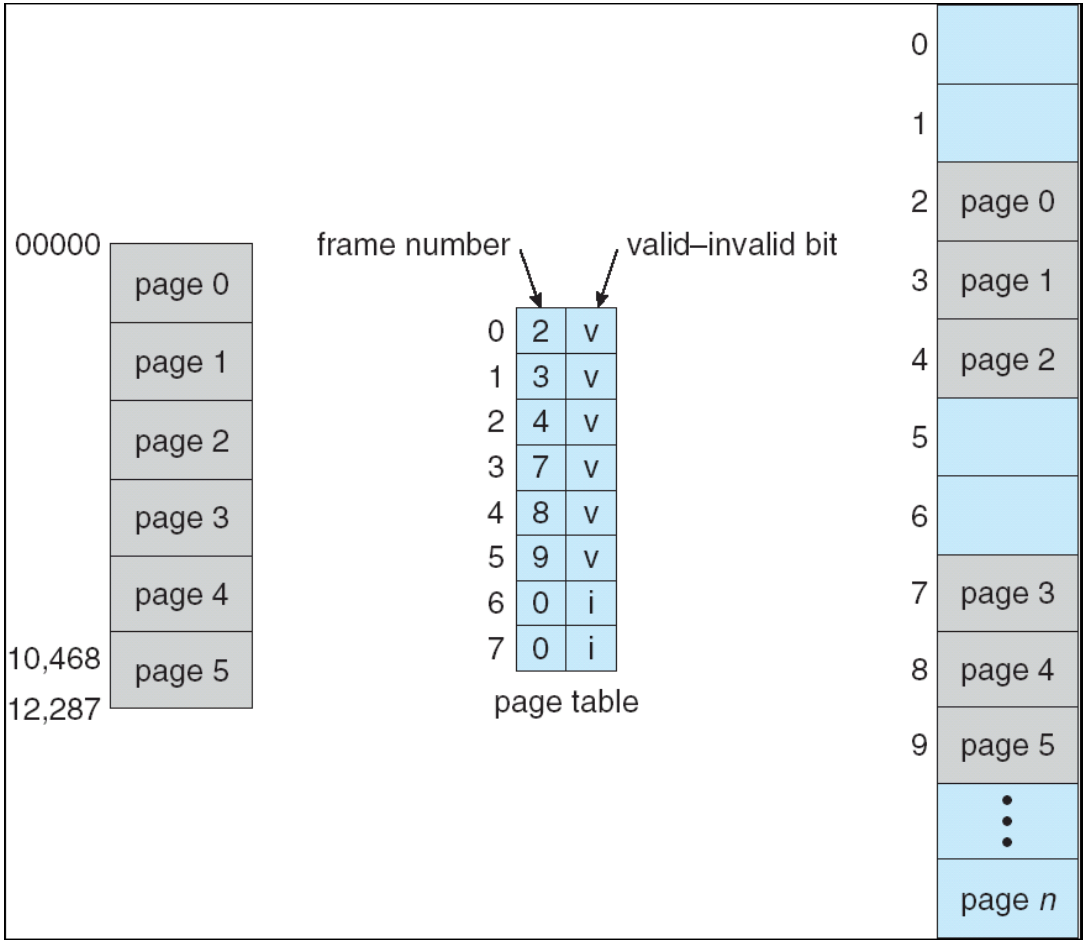
Hardware de paginação com TLB



Proteção de memória

- Proteção de memória implementada associando-se o bit de proteção a cada quadro
- Bit de **válido-inválido** anexado a cada entrada na tabela de página:
  - "válido" indica que a página associada está no espaço de endereço lógico do processo, e por isso é uma página válida
  - "inválido" indica que a página não está no espaço de endereço lógico do processo

Bit de válido-inválido na tabela de página

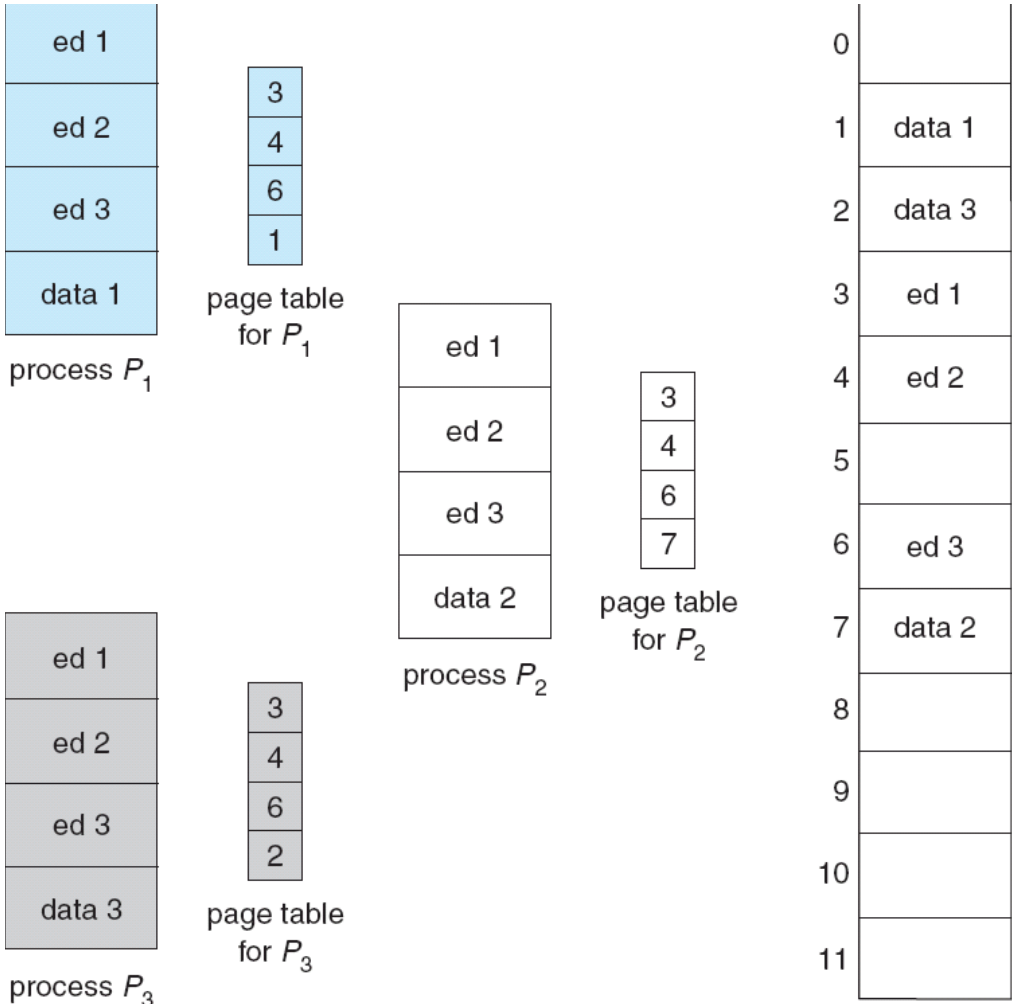


Páginas compartilhadas

- **Código compartilhado**
  - Uma cópia de código somente de leitura compartilhado entre processos (por exemplo, editores de texto, compiladores, sistemas de janela).
  - Código compartilhado deve aparecer no mesmo local no espaço de endereço lógico de todos os processos.

- **Código e dados privados**
  - Cada processo mantém uma cópia separada do código e dados
  - As páginas para o código e dados privados podem aparecer em qualquer lugar no espaço de endereço lógico

Exemplo de páginas compartilhadas



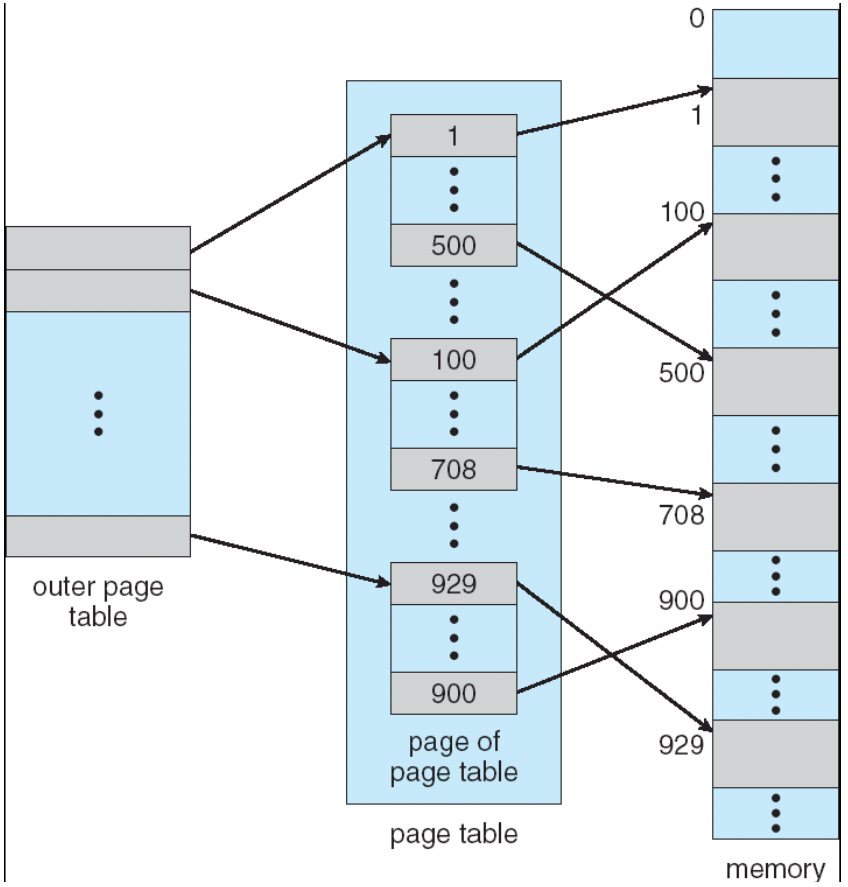
Estrutura da tabela de página

- Paginação hierárquica
- Tabelas de página com hash
- Tabelas de página invertidas

Tabelas de página hierárquicas

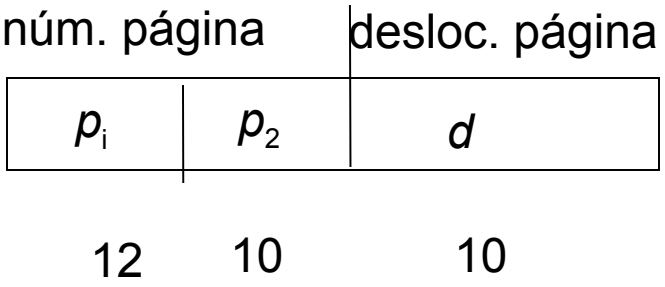
- Tabela de páginas pode ser muito grande, gerando novamente problema fragmentação da memória
- Solução: paginação da tabela de páginas
- Quebre o espaço de endereço lógico em múltiplas tabelas de página
- Uma técnica simples é uma tabela de página em dois níveis

Tabela de página em dois níveis



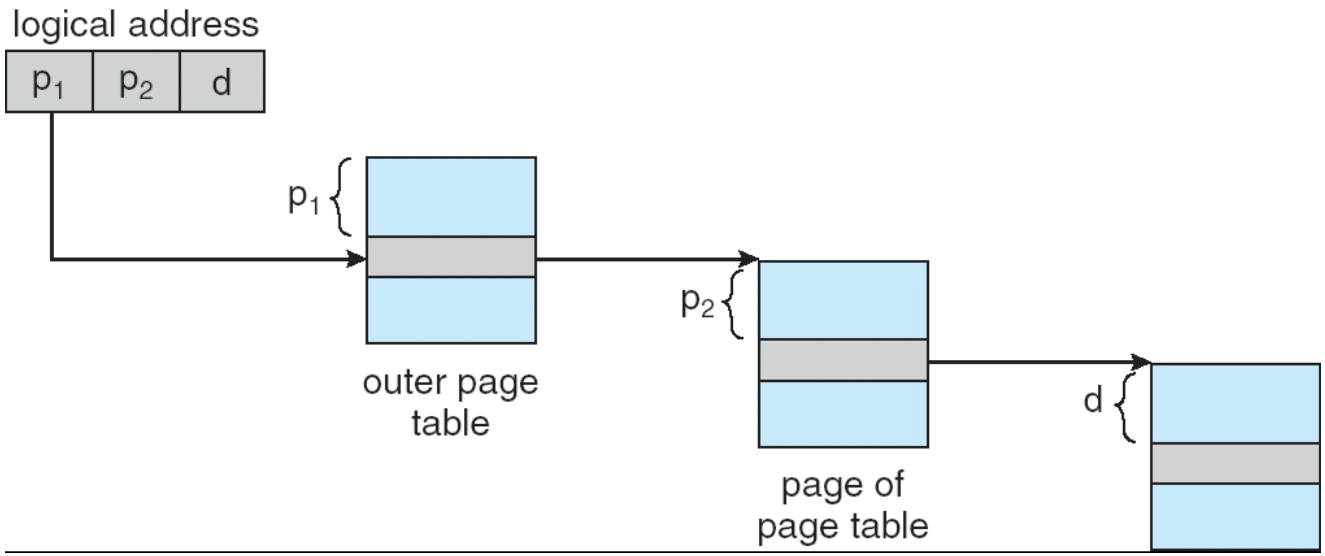
Exemplo de paginação em dois níveis

- Um endereço lógico (em máquinas de 32 bits com tamanho de página de 1K) é dividido em:
  - um número de página contendo 22 bits
  - um deslocamento de página contendo 10 bits
- Como a tabela de página é paginada, o número de página é dividido ainda em:
  - um número de página de 12 bits
  - um deslocamento de página de 10 bits
- Assim, um endereço lógico é o seguinte:



onde p_i é um índice para a tabela de página mais externa, e p_2 é o deslocamento da página dentro da tabela de página mais externa

Esquema de tradução de endereço



Esquema de paginação de três níveis

outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Tabelas de página em hash

- Para arquiteturas de 64 bits, as tabelas de página hierárquicas costumam ser impróprias (exigiria 7 níveis no UltraSPARC, por exemplo)
- Tabelas de página em hash são mais comuns em espaços de endereço > 32 bits (“diminuem o número de níveis”, num certo sentido)

Tabela de página em hash

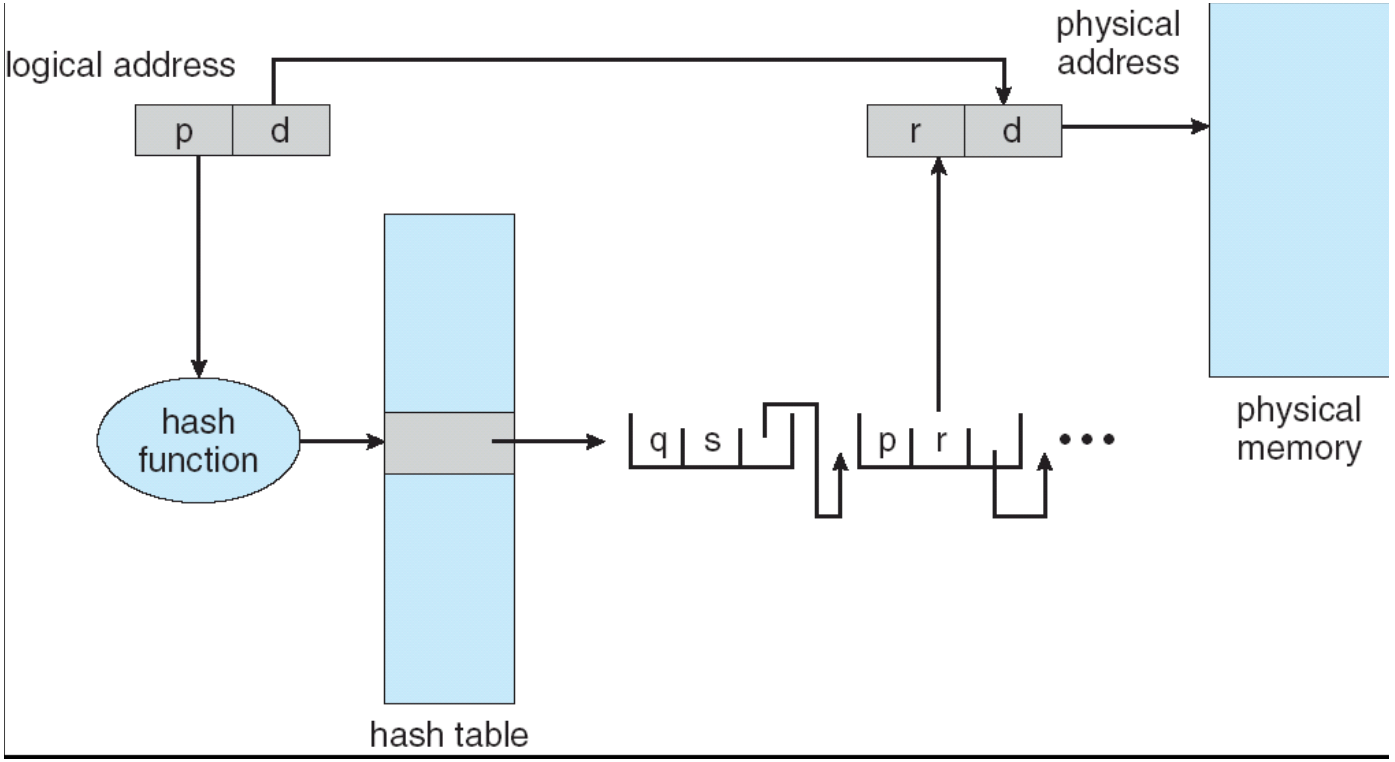
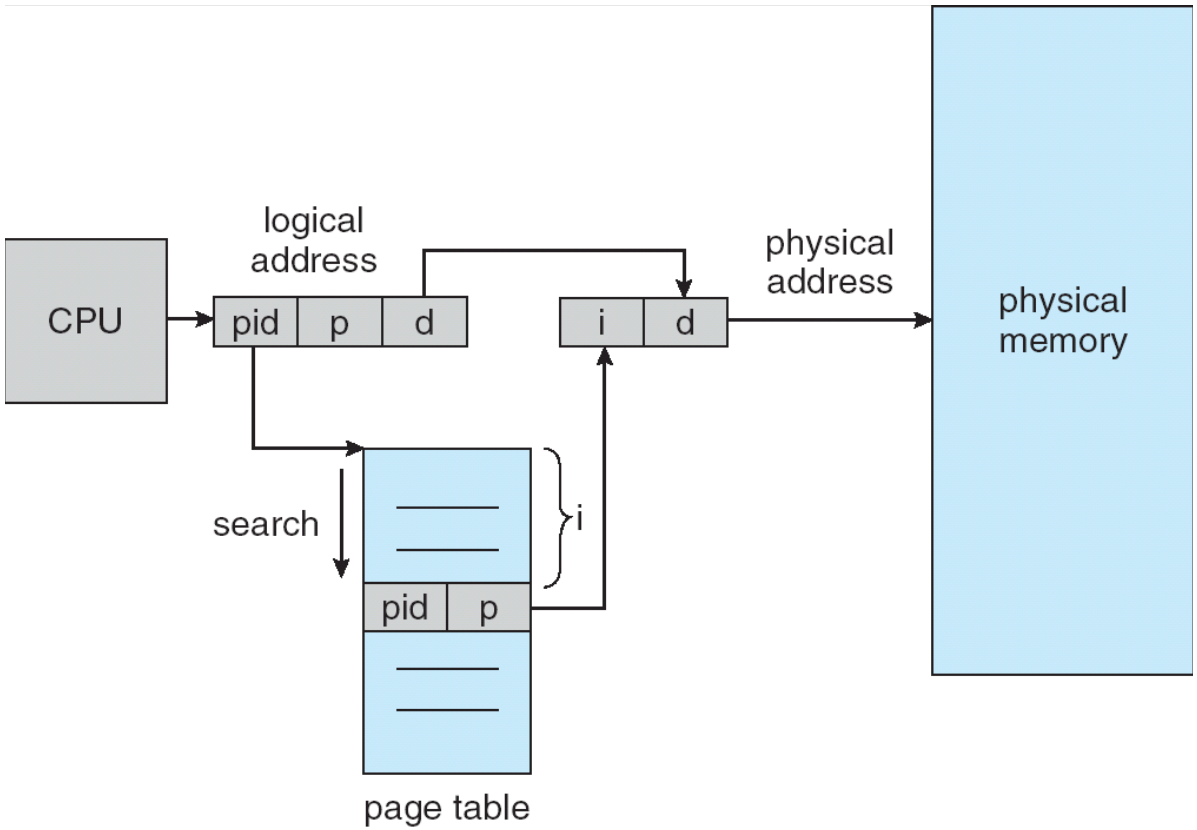


Tabela de página invertida

- Uma entrada para cada página real de memória
- Entrada consiste no endereço virtual da página armazenado nesse local da memória real, com informações sobre o processo que possui essa página
- Diminui a memória necessária para armazenar cada tabela de página (# de bits p/ representar pid + endereço virtual é menor do que # bits p/ representar endereço físico – comparar com slide 22), mas aumenta o tempo necessário para pesquisar a tabela quando ocorre uma referência de página
- Use tabela de hash para limitar a busca a uma ou, no máximo, algumas entradas de tabela de página

Arquitetura de tabela de página invertida



Segmentação

- Esquema de gerenciamento de memória que admite visão da memória pelo usuário
- Um programa é uma coleção de segmentos. Um segmento é uma unidade lógica como:

  programa principal,

  procedimento,

  função,

  método,

  objeto,

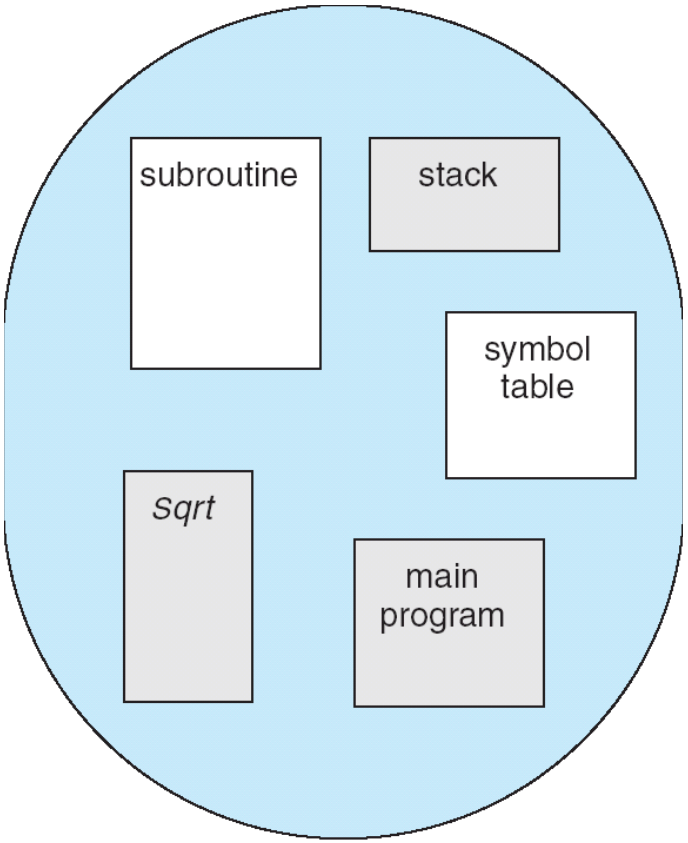
  variáveis locais, variáveis globais,

  bloco comum,

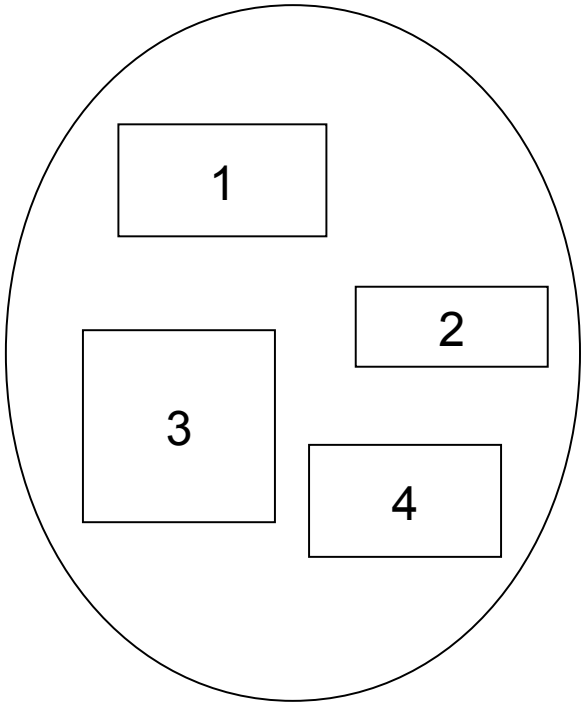
  pilha,

  tabela de símbolos, arrays

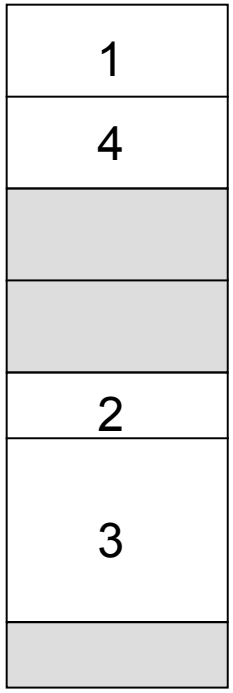
Visão de um programa pelo usuário



Visão lógica da segmentação



espaço do usuário



espaço da memória física

Arquitetura da segmentação

- Endereço lógico consiste em um par:

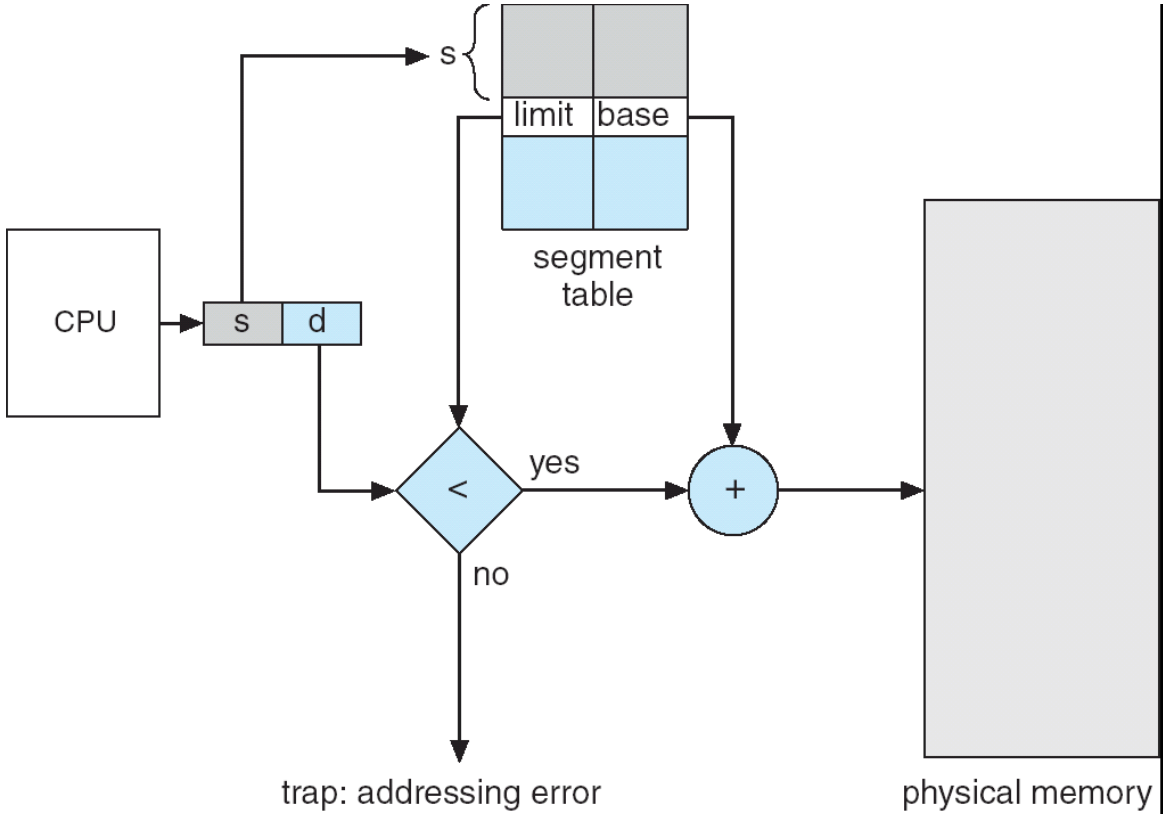
  <número-segmento, deslocamento>,

- **Tabela de segmento** – mapeia endereços físicos bidimensionais; cada entrada de tabela tem:
  - **base** – contém o endereço físico inicial onde os segmentos residem na memória
  - **limite** – especifica o tamanho do segmento

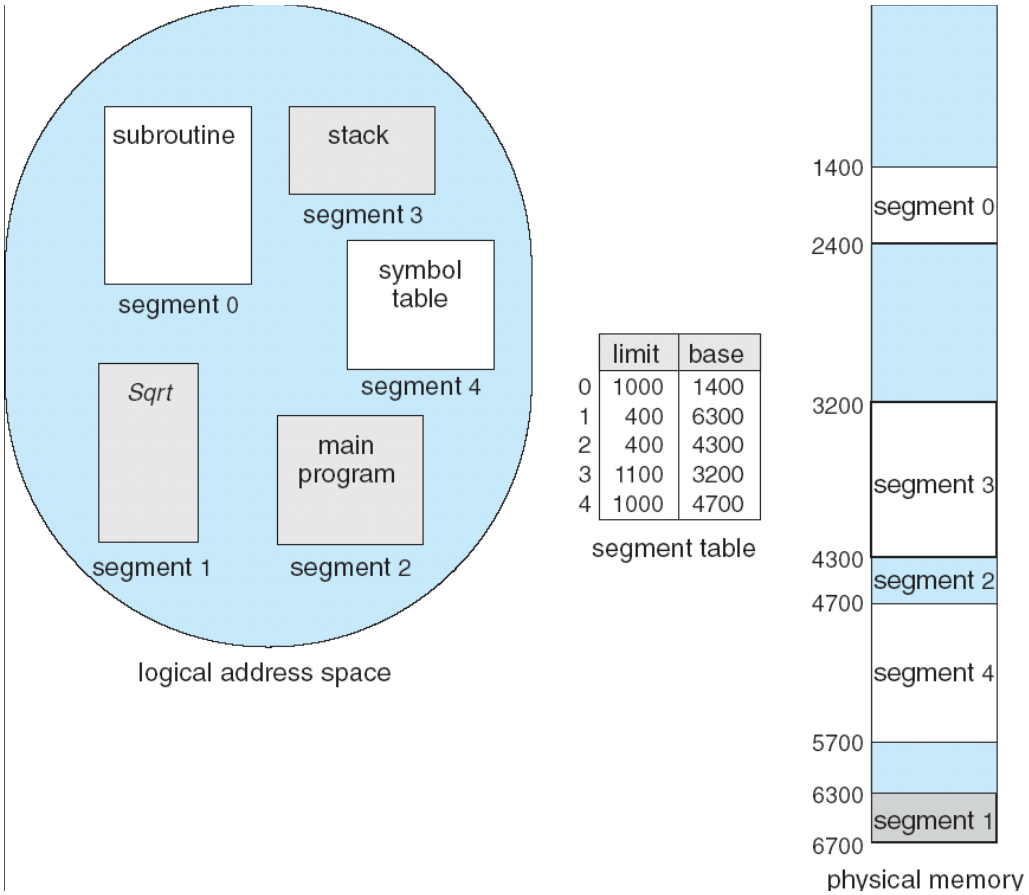
Arquitetura de segmentação (cont.)

- Proteção
  - A cada entrada na tabela de segmento, associe:
    - bit de validação = 0, então segmento ilegal
    - privilégios read/write/execute
- Bits de proteção associados a segmentos; o compartilhamento de código ocorre no nível de segmento
- Como os segmentos variam em tamanho, a alocação de memória é um problema de alocação dinâmica de armazenamento
- Um exemplo de segmentação aparece no diagrama a seguir

Hardware de segmentação



Exemplo de segmentação



Final do Capítulo 8