Gerência de Memoria

Referência:

Silberschatz, Abraham. Sistemas Operacionais com Java. 7 ed. Rio de Janeiro: Elsevier, 2008.

Fundamentos

Espaço de endereços lógico versus físico

Swapping

Alocação Contínua

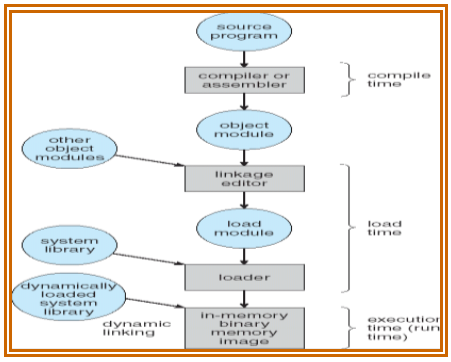
Paginação

Segmentação

Segmentação com paginação

Fundamentos

- Os programas precisam ser trazidos para a memória principal e colocados dentro de um processo para serem executados.
- *Fila de entrada* - coleção de processos no disco esperando para serem trazidos para a memória para a execução.
- Os programas usuários passam por diversas etapas antes de serem executados.



Definição dos endereços das instruções e dos dados na memória

- *No instante da compilação*: Se a posição na memória é conhecida a priori, códigos com endereços absolutos podem ser usados; O código terá de ser recompilado se o endereço inicial mudar.
- *No instante do carregamento*: Deve-se gerar código *relocável* se as posições de memória não são conhecidas na compilação.

- *No momento da execução*: A definição dos endereços deve ser atrasada até o momento da execução se o processo puder ser movido de um segmento de memória para outro. Precisa de suporte do hardware.

Carregamento dinâmico

- As rotinas não são carregadas na memória até que sejam chamadas.
- Melhor utilização do espaço de memória, rotinas não chamadas nunca são carregadas.
- Útil quando grandes quantidades de código são necessárias para processar casos raros.
- Nenhum suporte especial do sistema operacional é necessário, pode ser implementado pelo projeto do programa.

Ligação (edição de ligações) dinâmica

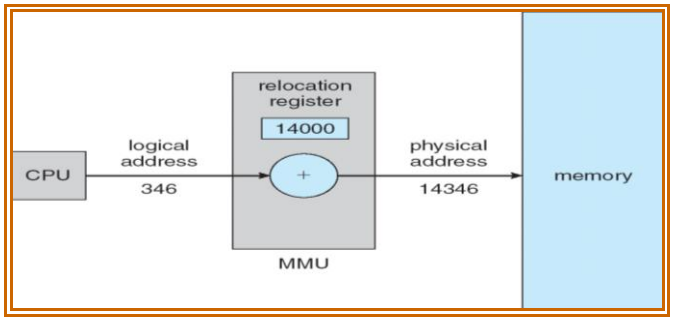
- As ligações (definição dos endereços das variáveis e das rotinas) é adiada até o momento da execução.
- Um pequeno trecho de código, chamado de *stub*, é usado para localizar a rotina verdadeira na memória.
- O *stub* substitui a si mesmo pelo endereço da rotina e executa a rotina.
- O sistema operacional precisa verificar se a rotina está no espaço de endereços do processo.

Overlays

- Mantem na memória apenas aquelas instruções que são necessárias num certo instante.
- Necessária quando o processo é maior do que a quantidade de memória alocada para ele.
- Implementada pelo usuário, nenhum suporte especial é necessário da parte do sistema operacional. O projeto da programação da estrutura de *overlays* é muito complexo.

Espaço de endereços lógico versus físico

- O conceito de um *espaço de endereços lógicos* que está associado a um *espaço de endereços físicos* é fundamental na gerência de memória.
    - *Endereço lógico* - gerado pela CPU; também chamado de endereço virtual.
    - *endereço físico* - usado para acessar a memória primária.
- Os endereços lógico e físicos são os mesmos nos esquemas de definição de endereços na compilação e no carregamento do programa; os endereços físicos e lógicos diferem no esquema de definição de endereços no momento da execução.



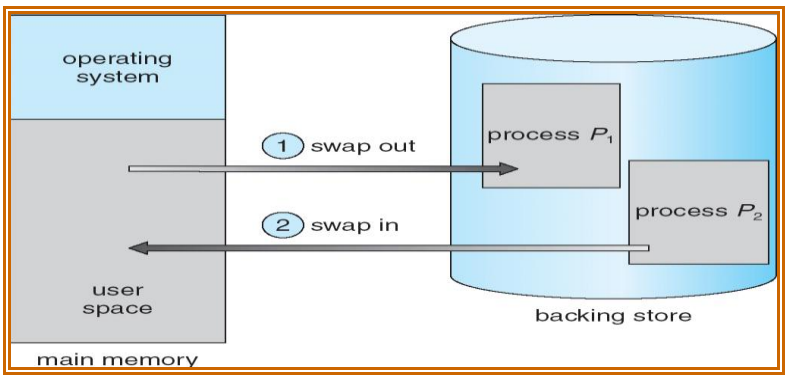
Tradução de endereço lógico em físico por registradores da MMU

Unidade de Gerência de Memória - MMU - *Memory Management Unit*

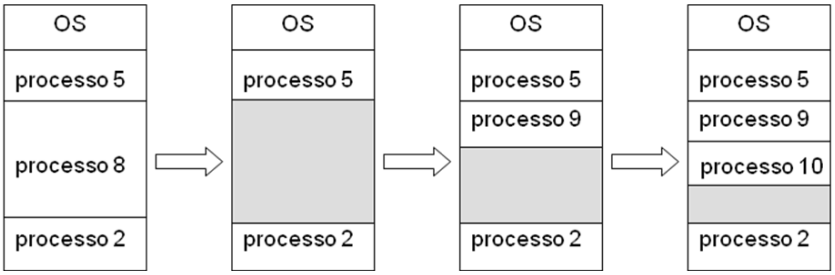
- Dispositivo em *hardware* que mapeia os endereços lógicos em físicos.
- No esquema acima da MMU, o valor do registrador de realocação é adicionado a todo endereço gerado por um processo usuário no momento em que ele é usado para acessar a memória.
- O programa usuário *lida* com endereços lógicos; ele nunca vê o endereço físico real.

Swapping

- Um processo pode ser *swapado* (*swapped*) temporariamente para fora da memória primária e levado para a memória secundária (área de *swap*, chamada pelo livro de *backing store*). Posteriormente, ele pode ser trazido de volta da área de *swap* para a memória para que o seu processamento continue.
- *Backing store* - disco rápido grande o suficiente para acomodar cópias de todas as imagens de memória de todos os usuários; deve fornecer acesso direto a estas imagens de memória.
- *Roll out, roll in* - versão de *swapping* usada em algoritmos de escalonamento baseados em prioridade; processo de menor prioridade é *suapado* para que um de maior prioridade possa ser carregado e executado.
- A maior parte do tempo de *swap* é dada pelo tempo de transferência; o tempo total de transferência é diretamente proporcional à quantidade de memória *suapada*.
- Versões modificadas de *swapping* são encontradas na maioria dos sistemas, p.ex., UNIX e MS Windows.

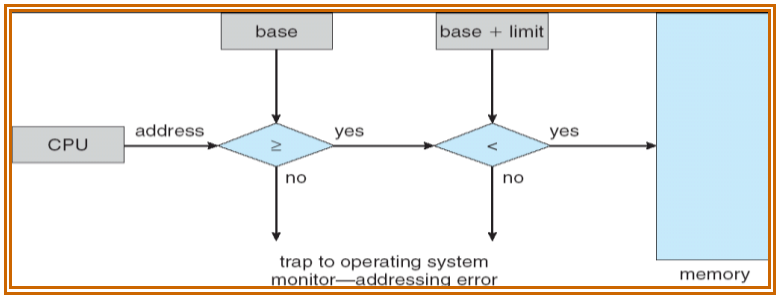


Alocação Contínua (contígüa)



Carregamento de programas/processos.

- A memória principal é geralmente dividida em duas partições:
  - Parte residente do sistema operacional, geralmente nos endereços mais baixos de memória juntamente com o vetor de interrupções.
  - Parte dos processos de usuários usando os endereços mais altos da memória.



Hardware de checagem de limites de memória

- Alocação com uma única partição:
  - O esquema com o registrador de realocação é usado para proteger os processos usuários uns dos outros, além de proteger os sistema operacional de mudanças no seu código ou nos seus dados.
  - O registrador de relocação contem o menor endereço físico que o processo pode acessar; o registrador *limit* verifica se os endereços são menores do que o máximo permitido.

- Alocação com múltiplas partições:

  - *Buraco* - Bloco de memória disponível; buracos de tamanhos variados podem estar espalhados pela memória.
  - Quando um processo chega, ele recebe a sua memória de um buraco grande o bastante para acomodá-lo. O sistema operacional mantem informações a respeito de:
    1. partições alocadas
    2. partições livres

Problema da alocação de espaço de armazenamento dinâmico

- *First fit* (o primeiro adequado): Aloca o primeiro buraco grande o suficiente.
- *Best fit* (o mais adequado): aloca o *menor* buraco grande o suficiente; Deve procurar em toda a lista de espaços livres, a menos que ela esteja ordenada por tamanho. Produz o menor buraco após a ocupação.
- *Worst fit* (o menos adequado): Aloca o maior buraco; de novo, deve-se procurar em toda a lista. Produz o maior buraco depois de uma alocação.

Fragmentação

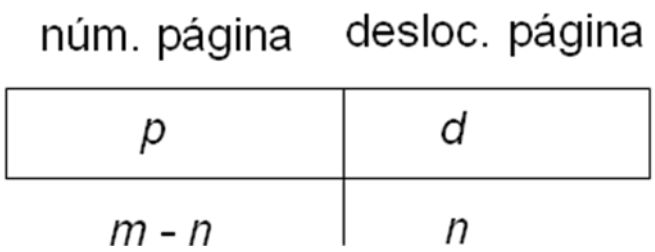
- **Fragmentação externa** - Existe memória livre o suficiente para atender uma requisição, mas ela não é contigüa.
- **Fragmentação interna** - A memória alocada pode ser ligeiramente maior do que a requisitada; a diferença se deve ao tamanho fixo das partições.
- Redução da fragmentação por compactação:
    - Deslocar o conteúdo da memória para que a área de memória livre fique num único bloco.
    - A compactação só é possível se existir um mecanismo dinâmico de *relocação*, já que ela é feita em tempo de execução.
    - problema das Entradas e Saídas:
        - o processo espera na memória enquanto está envolvido em entradas e saídas.
        - As entradas e saídas só podem ser feitas através dos *buffers* do sistema operacional.

Paginação

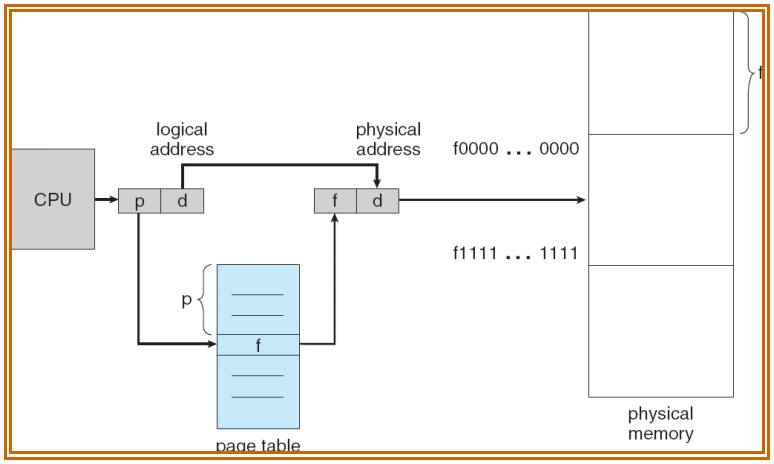
- O espaço lógico de endereços pode não ser contigüo; o processo recebe alocação de memória física sempre que houver disponibilidade.
- Divide a memória física em blocos de tamanho fixo chamados de quadros (os tamanhos são sempre uma potência de 2, entre 512 e 64k).
- Divide a memória lógica em blocos de mesmo tamanho chamados de *páginas*.
- Guarda-se a informação sobre todos os quadros livres.
- Para rodar um programa que precisa de *n páginas*, é necessário ter-se *n quadros livres* e carregar o programa neles.
- Uma *tabela de páginas* é usada para traduzir um endereço lógico em físico.
- Pode existir fragmentação interna.

- O endereço gerado pela CPU é dividido em:

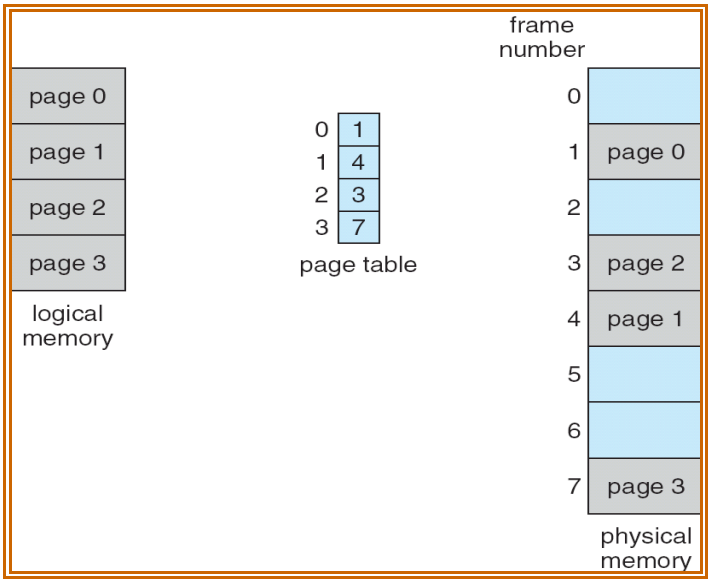
    - **Número de página (*p*)** – usado como um índice para uma *tabela de página* que contém endereço de base de cada página na memória física
    - **Deslocamento de página (d)** – combinado com endereço de base para definir o endereço de memória físico que é enviado à unidade de memória
    - Para determinado espaço de endereço lógico 2^m e tamanho de página 2^n



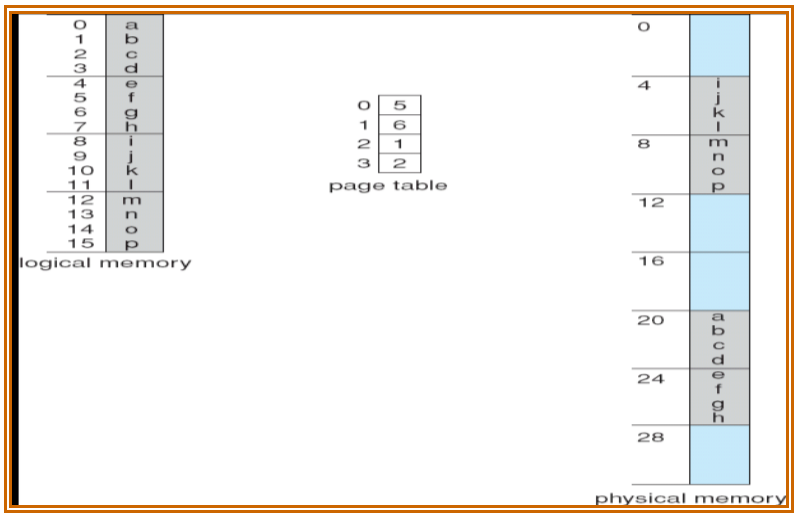
Hardware de paginação



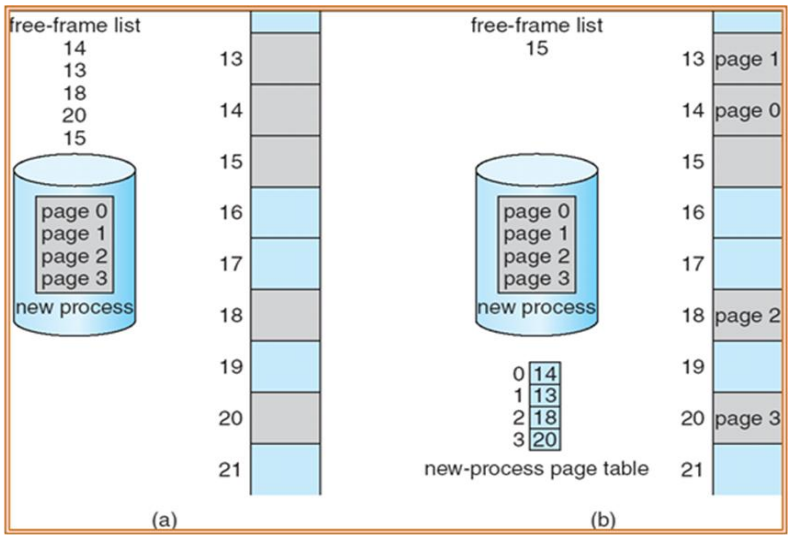
Modelo de paginação da memória lógica e física



Exemplo de paginação



Quadros livres



Antes da alocação Após a alocação

Esquema de tradução

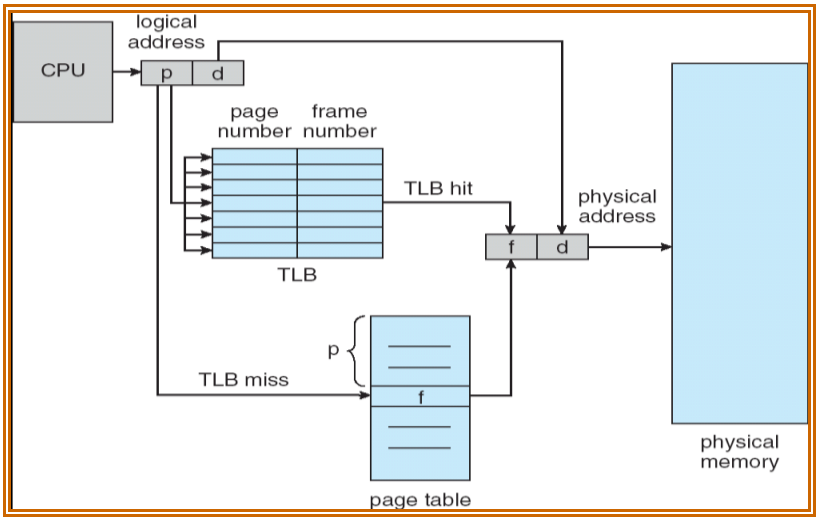
Tradução endereço lógico em endereço físico

O endereço gerado pela CPU é dividido em duas partes:

1. *Número de página (p)* - usado como um índice na tabela de páginas que contém o número do quadro usado para guardar esta página.
2. *Deslocamento na página (d)* - combinado com o número do quadro, define o endereço físico da posição de memória desejada.

Implementação da tabela de páginas

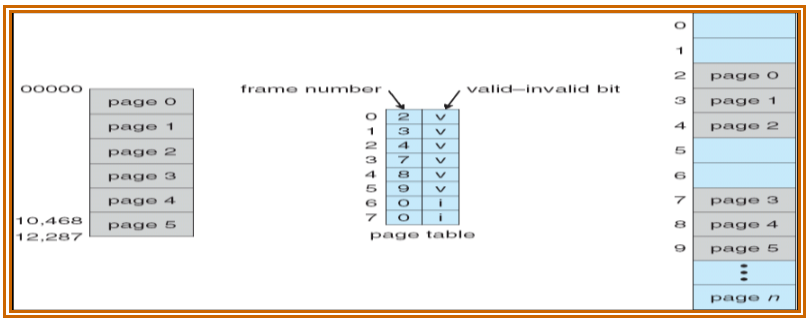
- A tabela de páginas é guardada na memória principal.
- O *PTBR (Page-table base register)* aponta para a tabela de páginas.
- O *PRLR (Page-table length register* indica o tamanho da tabela de páginas.
- Neste esquema, cada acesso a instruções, ou dados, requer 2 acessos à memória. Um para obter o número do quadro pela tabela de páginas e outro para obter a instrução, ou dado.
- O problema de 2 acessos pode ser resolvido com o uso de uma cache em *hardware* especializada usando memória associativa, chamada normalmente de *TLB - Table Lookaside Buffer*.



Uso da TLB

Proteção da memória

- A proteção da memória é obtida com a associação de bits de proteção para cada quadro da memória.
- Um *bit válido-inválido* é colocado em cada linha da tabela de páginas:
  - *válido* indica que a página associada está no espaço lógico de endereços e, portanto, é uma página *legal* do processo.
  - *inválido* indica que a página não está no espaço lógico do processo.



Bit de validade

Uso de dois níveis para a tabela de páginas

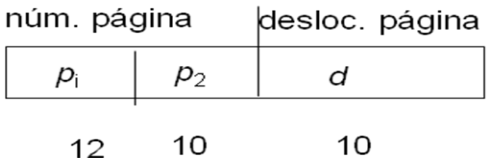
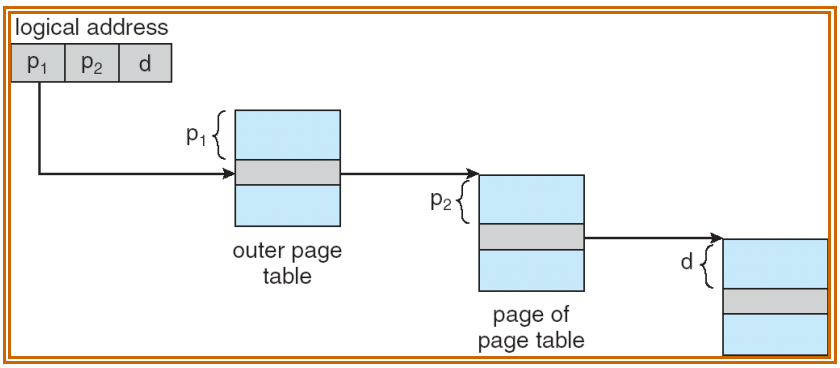


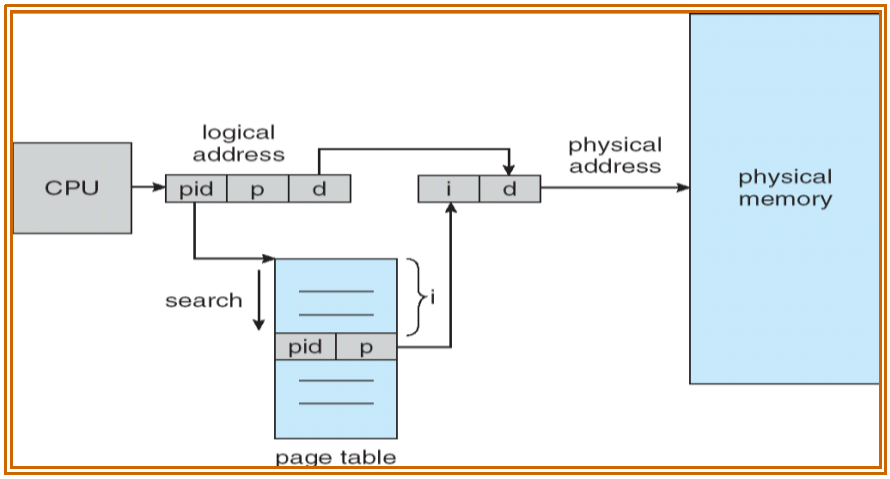
Tabela de páginas de 2 níveis



Tradução com 2 tabelas de páginas

Tabela de páginas invertida

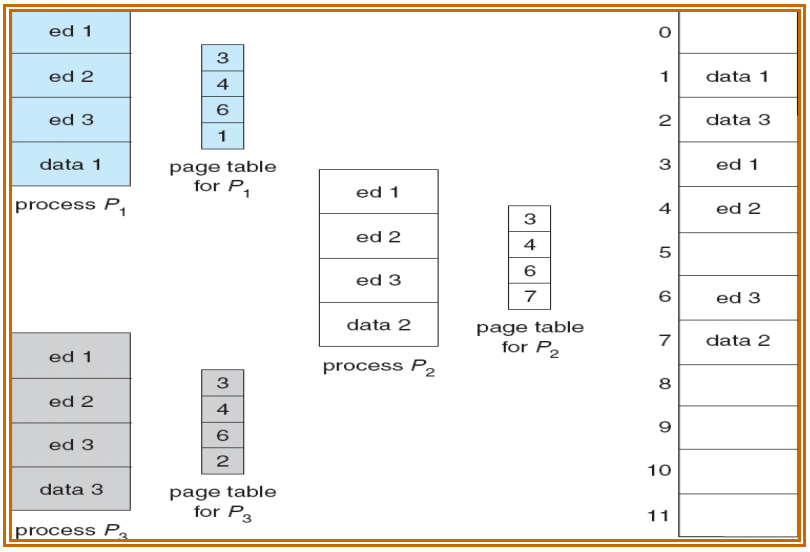
- Uma linha para cada quadro de memória
- O conteúdo da linha consiste da página virtual e do processo que possui este quadro
- Diminui a memória necessária para guardar cada tabela de páginas mas aumenta o tempo de busca na tabela quando uma referência a uma página ocorre.
- Usa tabela *hash* para limitar a busca a uma ou no máximo umas poucas entradas na tabela de páginas.



Tradução com tabela de páginas invertida

Compartilhamento de páginas

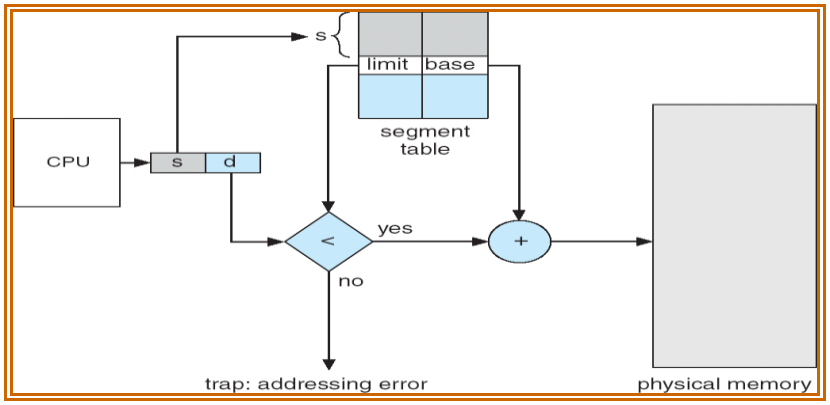
- Código compartilhado
  - Uma cópia de código apenas-leitura (reentrante) entre os processos.
  - O código compartilhado deve aparecer no mesmo endereço lógico de todos os processos.
- Código e dados privados
  - Cada processo guarda uma cópia separada do código e dos dados.
  - As páginas de código e de dados privadas podem estar em qualquer lugar no espaço de endereços lógicos.



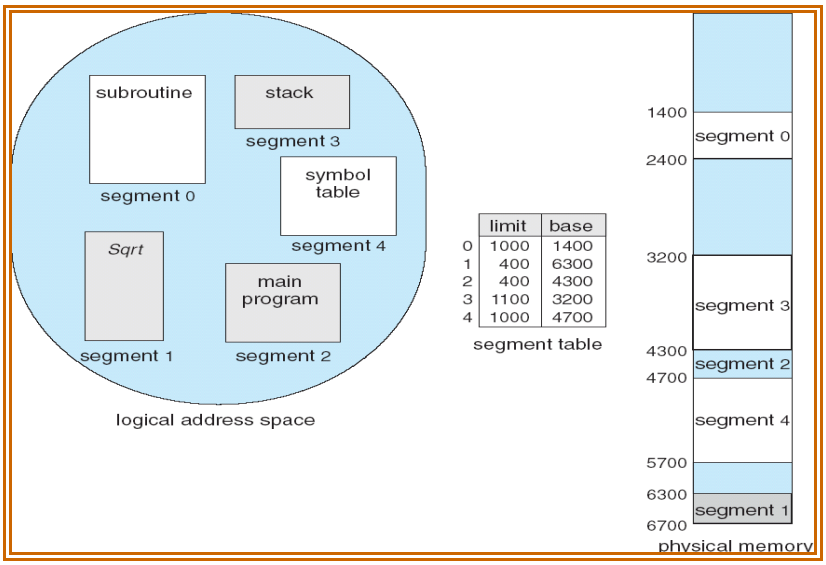
Compartilhamento de páginas entre processos

Segmentação

- Esquema de gerência de memória que suporta visões do usuário da memória.
- Um programa é uma coleção de segmentos. Um segmento é uma unidade lógica tal como:
  - programa principal
  - rotina
  - função
  - variáveis locais
  - variáveis globais
  - bloco comum
  - pilha
  - tabela de símbolos
  - vetores



Segmentação e limites válidos de memória



Exemplo de Segmentação