Gerenciamento de Memória

Conceitos básicos

Swapping

Alocação contígua

Paginação

Segmentação

Segmentação com paginação

Gisele S. Craveiro

Conceitos básicos

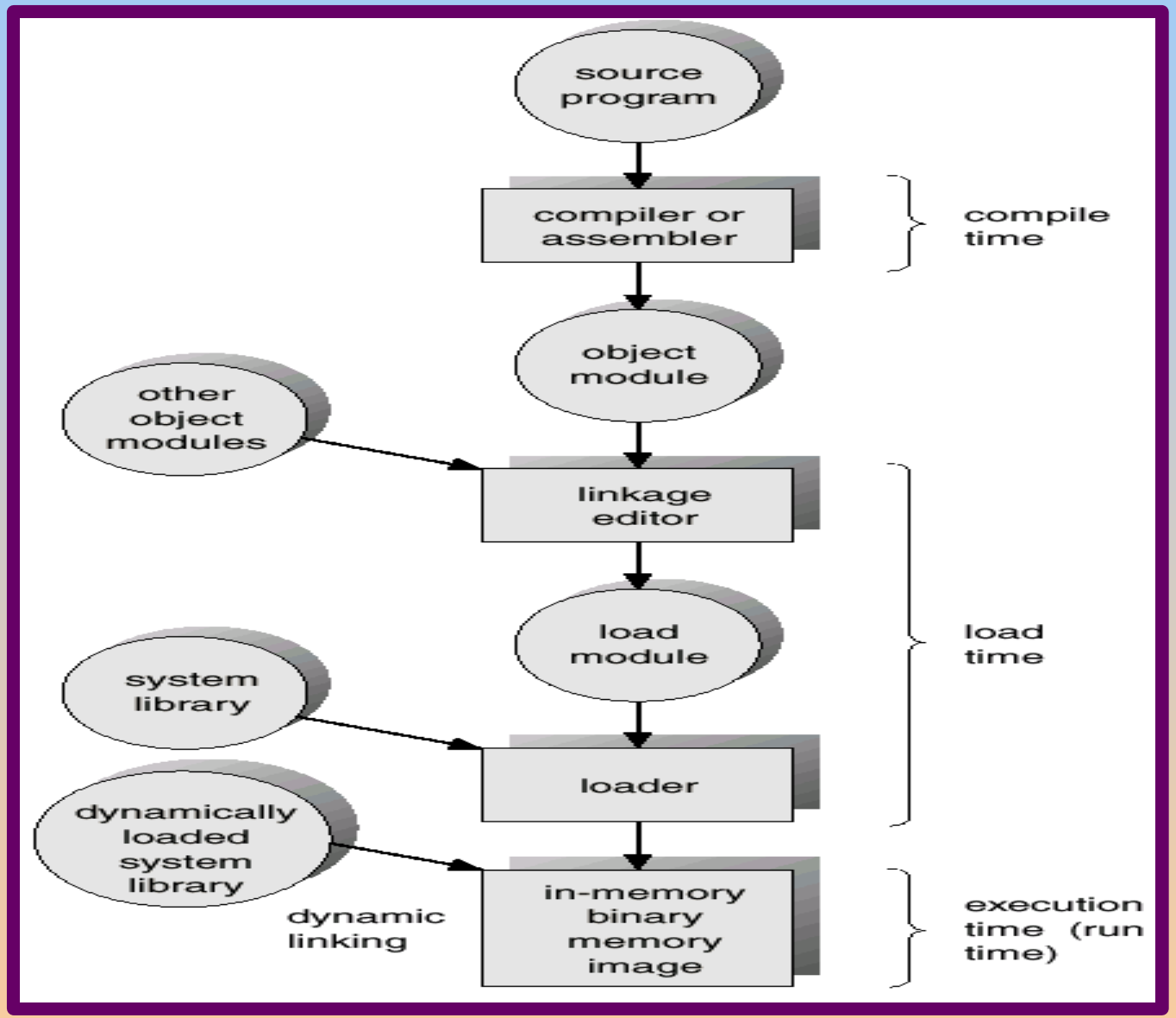
- Um programa precisa ser alocado na memória e associado a um processo para que possa executar.
- *Fila de entrada* – coleção de processos que estão no disco esperando ser colocados na memória para que o programa seja executado.
- Programas passam por vários processos até se tornarem executáveis.

Vinculação de código e dados na memória

Vinculação de endereços de instruções e dados para endereços de memória podem ser feitos em três diferentes estágios:

- **Tempo de compilação**: Se a localização na memória já é conhecida a priori, um código de endereçamento absoluto por ser gerado. É necessário recompilar o código se o endereço inicial muda.
- **Tempo de carregamento**: Precisa gerar código relocável se a localização de memória não é conhecida em tempo de compilação.
- **Tempo de execução**: Vinculação é atrasada até o tempo de execução se o processo pode mover durante sua execução de um segmento de memória para outro. Necessita de hardware para suportar mapeamento de endereço como, por exemplo, registradores base e limite.

Processamento de um programa



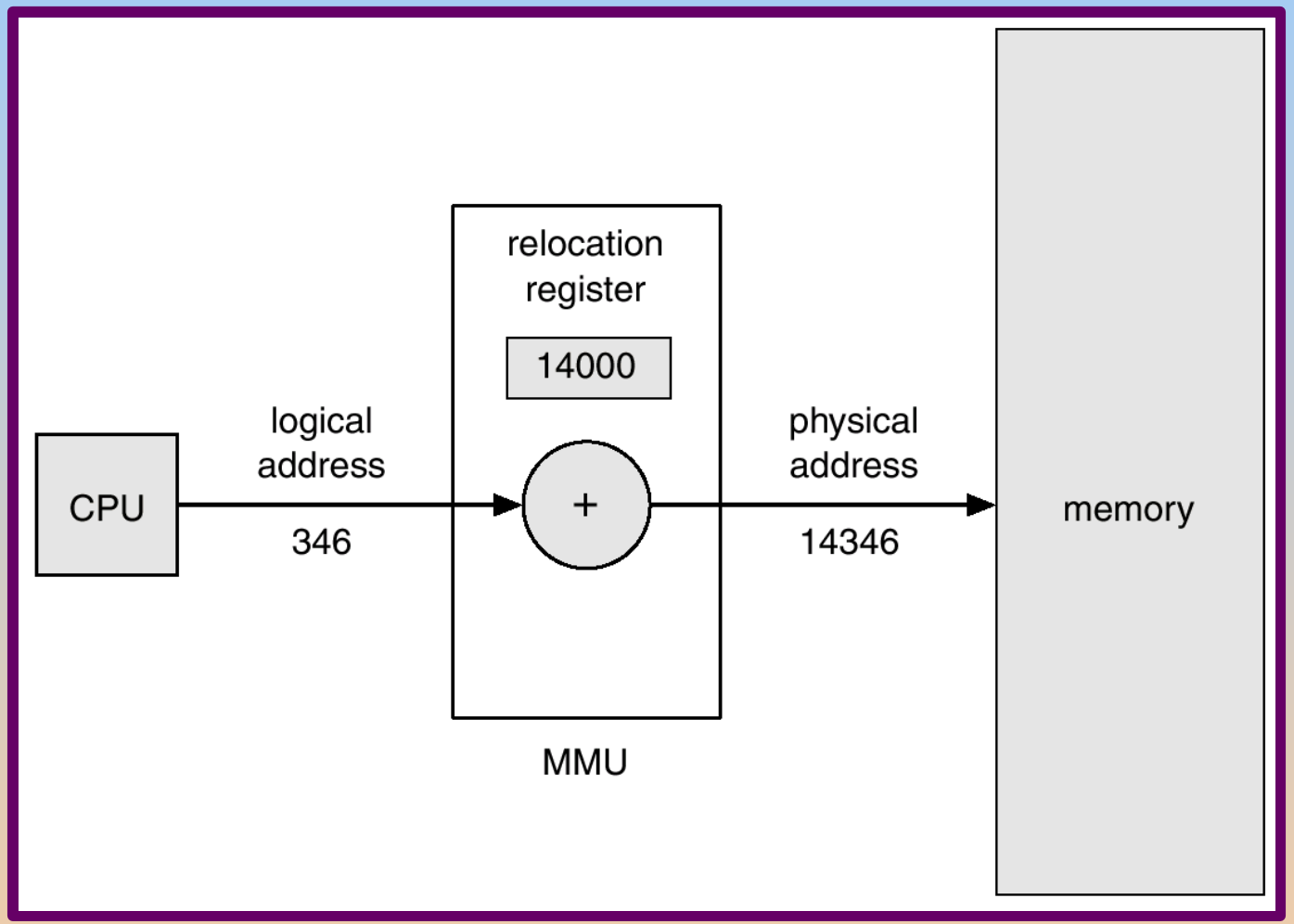
Espaços de endereçamento lógico e físico

- A separação entre espaço de endereçamento lógico e físico é fundamental para gerenciamento de memória.
  - *Endereço lógico* – gerado pela CPU; também conhecido como *endereço virtual*.
  - *Endereço físico* – endereço visto por uma unidade de memória.
- Endereços lógicos e físicos são os mesmos em esquemas de vinculação de endereços em tempo de compilação e de carregamento; porém, endereços lógicos (virtuais) e endereços físicos diferem nos esquemas de vinculação de endereço em tempo de execução.

Memory-Management Unit (MMU)
Unidade de gerenciamento de memória

- Dispositivo de hardware que mapeia endereços virtuais em físicos.
- Dentro do esquema da MMU, o valor de um registrador de relocação é adicionado a todo endereço gerado pelo processo de um usuário, antes de ser enviado para a memória.
- O programa do usuário lida sempre com endereços lógicos, ele nunca vê endereços físicos.

Relocação dinâmica usando um registrador de relocação



Carregamento dinâmico

- Rotina não é carregada até que ela seja invocada
- Consiste numa utilização mais otimizada de memória; rotinas não utilizadas nunca são carregadas.
- Útil quando grandes quantidades de código são necessários para manipular casos pouco frequentes.

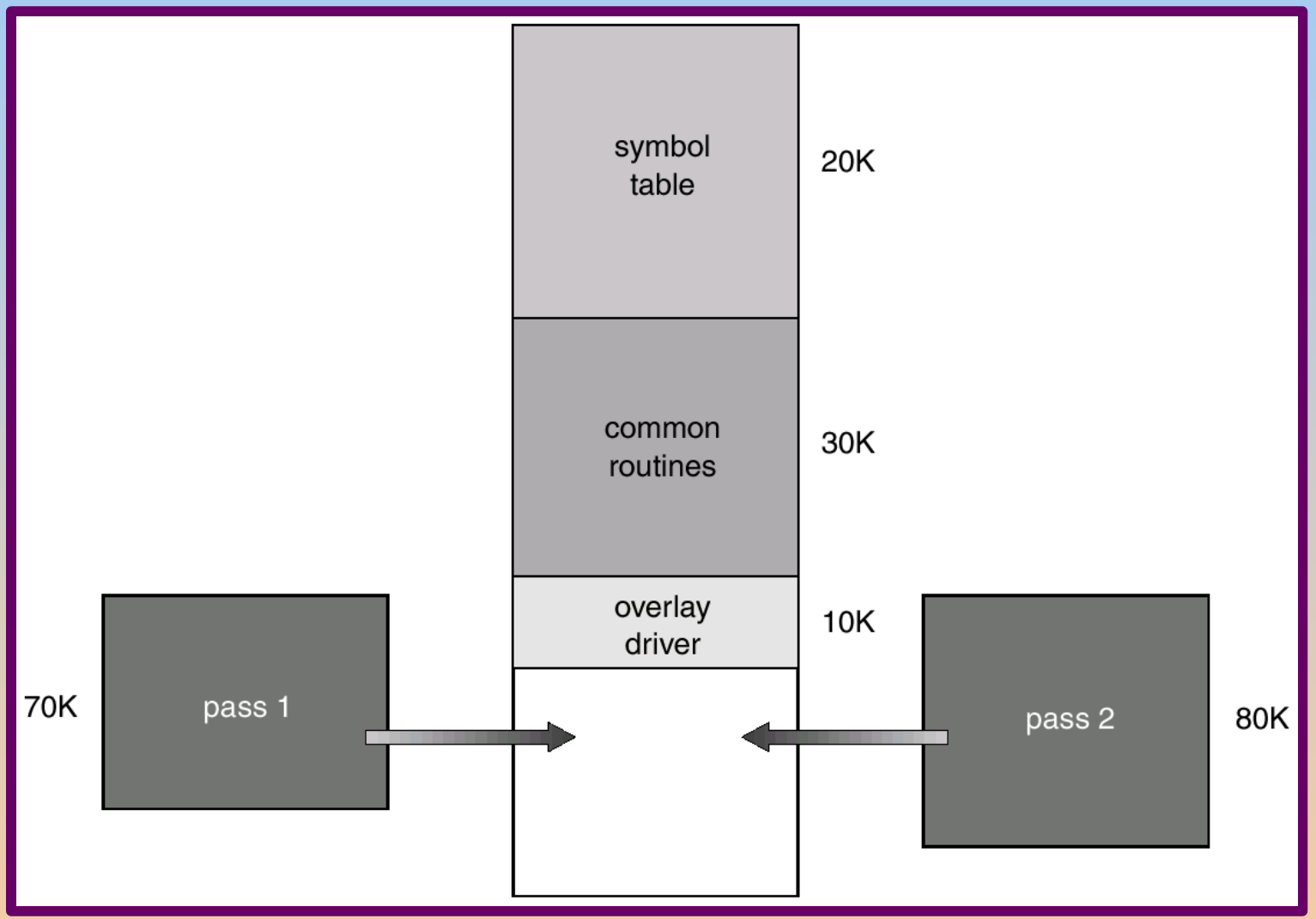
Ligação dinâmica

- Ligação é adiada até o tempo de execução.
- Pequeno pedaço de código, o stub, é usado para localizar a apropriada rotina, residente na memória.
- O stub substitui a si mesmo com o endereço da rotina e a executa.
- O sistema operacional necessita apenas checar se a rotina está no espaço de endereçamento do processo.
- Ligação dinâmica é particularmente útil para bibliotecas.

Overlays(sobreposições)

- Manter na memória somente quelas instruções e dados que são necessários num determinado instante.
- Necessárias quando um processo é maior que a quantidade de memória alocada para ele.
- Implementada pelo usuário, sem necessidade de suporte especial pelo sistema operacional. O projeto da estrutura de overlay é complexo.

Overlays para um montador de dois passos

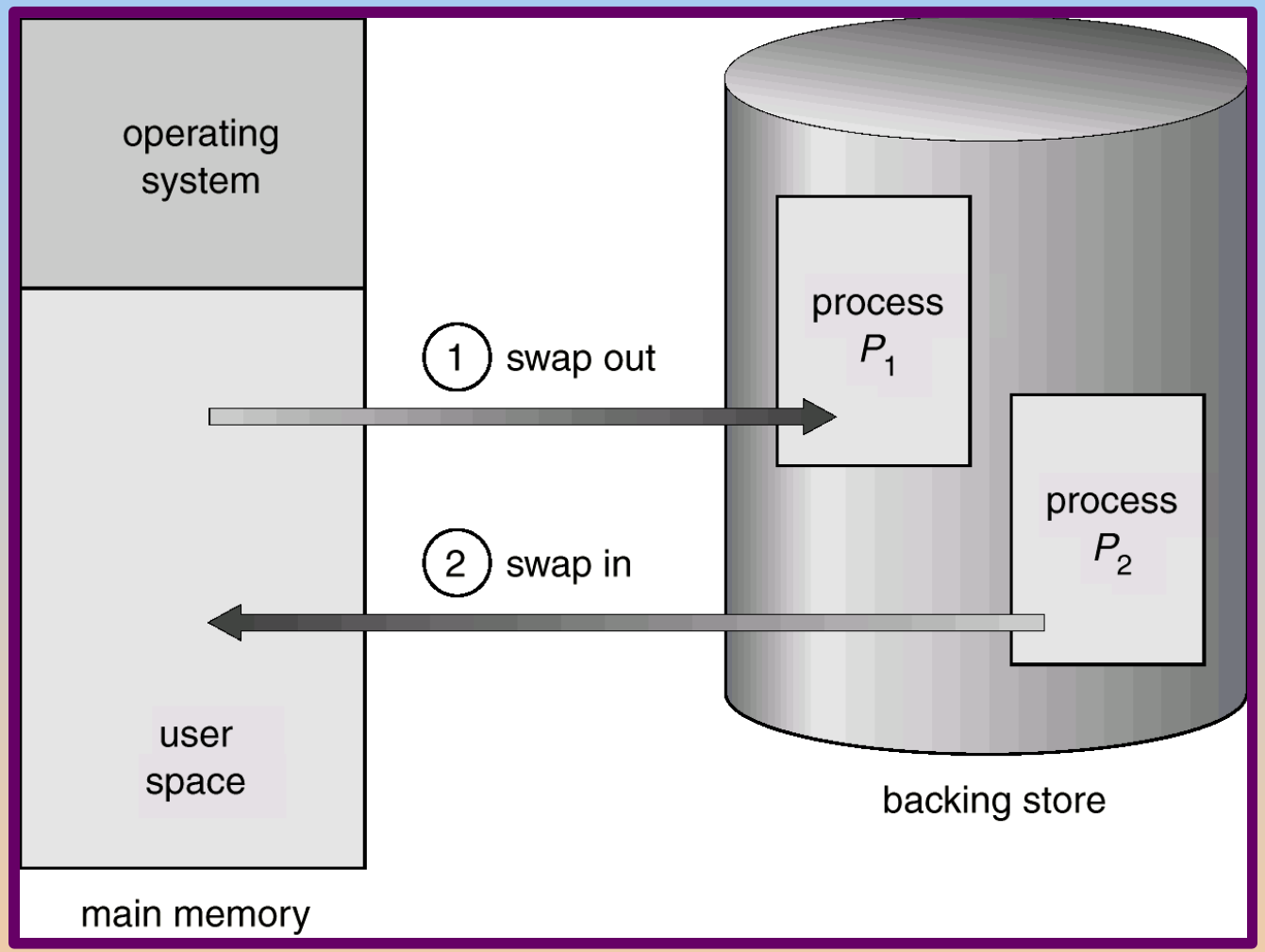


Swapping

- Um processo pode ser trocado(swapped) temporariamente da memória para um meio de armazenamento(Backing store), e reconduzido à memória para posterior execução.

- Backing store – um disco rápido e grande o suficiente para acomodar cópias de todas as imagens de memória de todos os utuários; precisa disponibilizar acesso direto a todas estas imagens de memória.
- R*oll out, roll in* – variante de swapping usada por algoritmos de escalonamento baseados em prioridade; processo de baixa prioridade são amrzenados no disco para que outros processo possam ser carregados e executados.
- O maior tempo do processo de troca é devido à transferências disco-memória; tempo total de transferência é diretamente proporcional à quantidade de memória sendo armazenada no disco;

Visão esquemática de Swapping

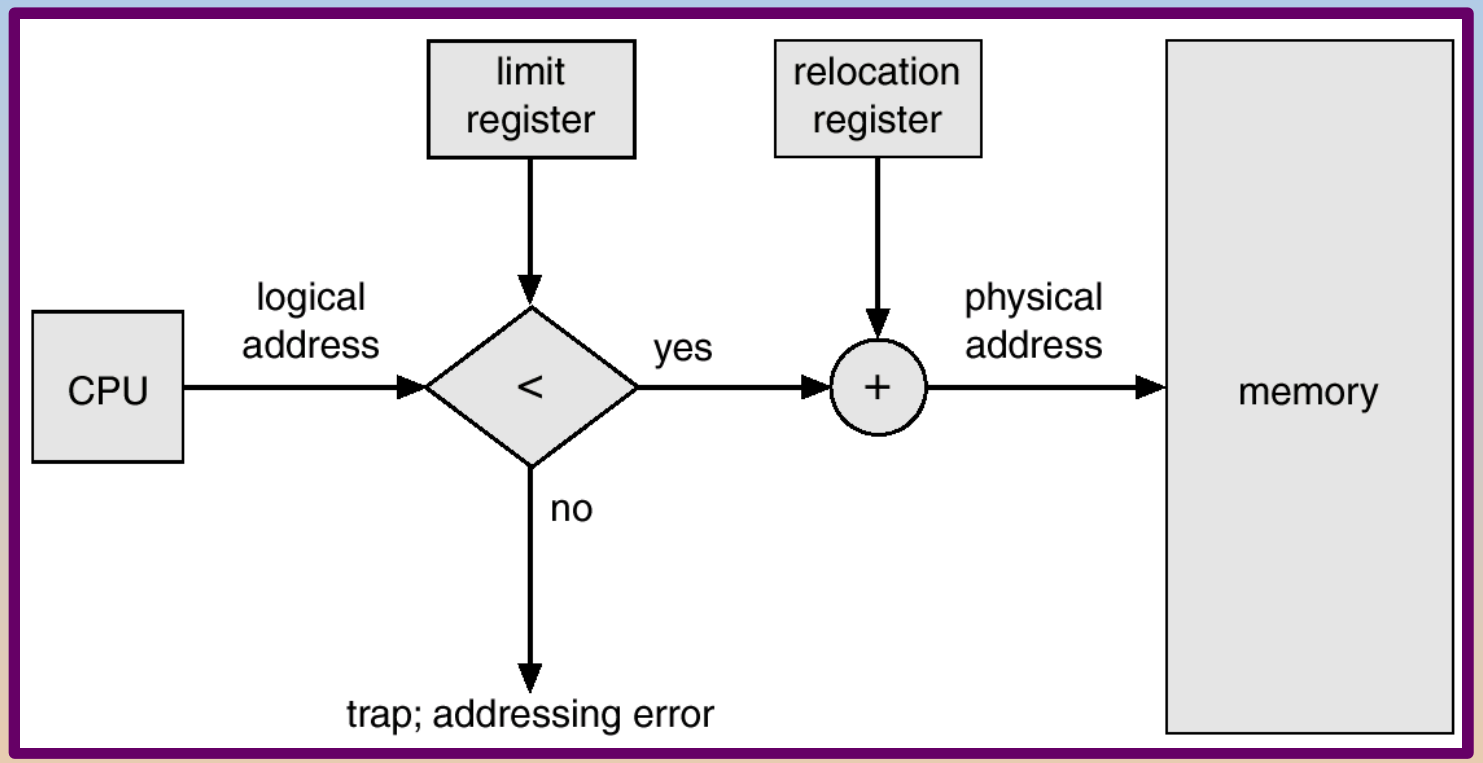


Alocação contígua

- Memória principal é vista como duas partições:
  - Sistema operacional residente, usalmente carregado na memória baixa juntamente com o vetor de interrupções.
  - Processos do usuário, armazenados na área restante.

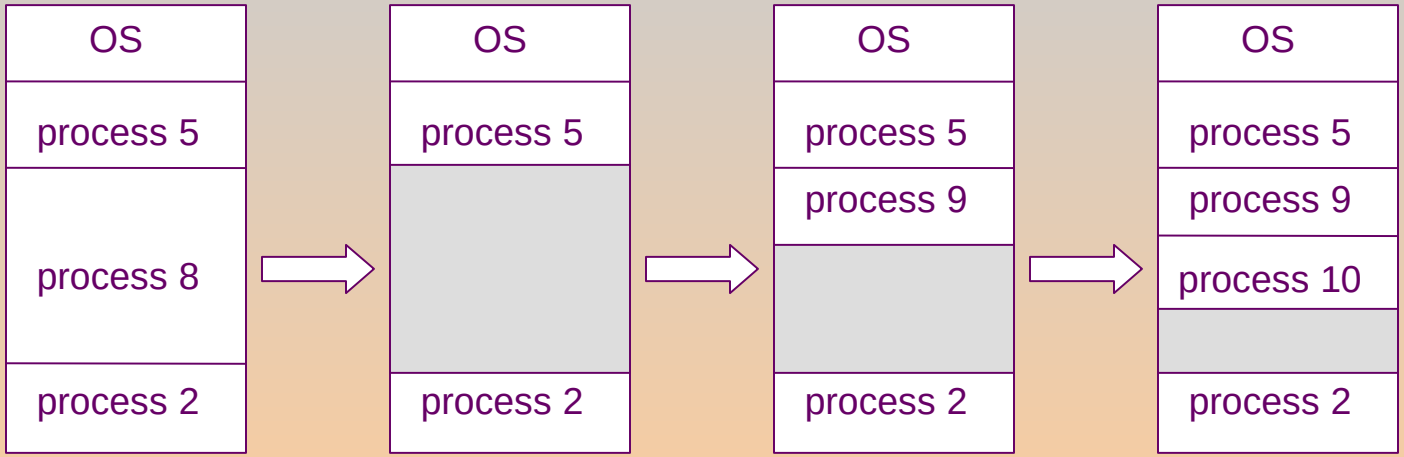
- Alocação com partição simples:
  - Esquema de registrador de relocação usado para proteger processos de um usuário de outros peocessos, e também do código e dados dos sistema operacional.
  - Registrador de relocação contém valor do menor endereço físico; registrador-limite contém o intervalo de endereços lógicos – cada endereço lógico precisa ser menor que o registrador limite.

Suporte de hardware para registradores de relocação e limite



Alocação contígua (Cont.)

- Alocação com múltiplas partições
  - *Buraco* – bloco de memória disponível; buracos de vários tamanhos estão distribuídos através da memória.
  - Quando um processo chega, ele é alocado em um buraco grande o suficiente para cabê-lo.
  - Sistemas operacional mentém informação sobre: a) partições alocadas b) partições livres (buracos)



Problema da alocação de partições

Como satisfazer uma requisição de tamanho n de uma lista de partições livres?

- **First-fit**: alocar a primeira partição livre que acomoda o processo.
- **Best-fit**: Aloca a menor partição dentre as que podem acomodar o processo; precisa procurar numa inteira de partições livres.
- **Worst-fit**: Aloca a maior partição dentre as que podem acomodar o processo; precisa, também, procurar numa inteira de partições livres.

First-fit e best-fit são melhores que worst-fit em termos de velocidade e armazenamento, respectivamente.

Fragmentação

- **Fragmentação externa** – um total de memória existe para satisfazer uma requisição, mas ela não é contígua.
- **Fragmentação interna** – memória alocada pode ser levemente maior que a memória requisitada; esta diferença é memória interna de uma partição, mas não está sendo usada.
- Reduzir a fragmentação externa por compactação:
  - Reorganizar o conteúdo de memória para colocar toda a memória livre num único bloco.
  - Compactação de memória é possível somente quando a relocação é dinâmica, e é feita em tempo de execução.
  - Problema de E/S:
    - ✔ Armazenar uma tarefa em memória enquanto ela está envolvida em E/S.
    - ✔ Realize E/S usando somente buffers do SO.

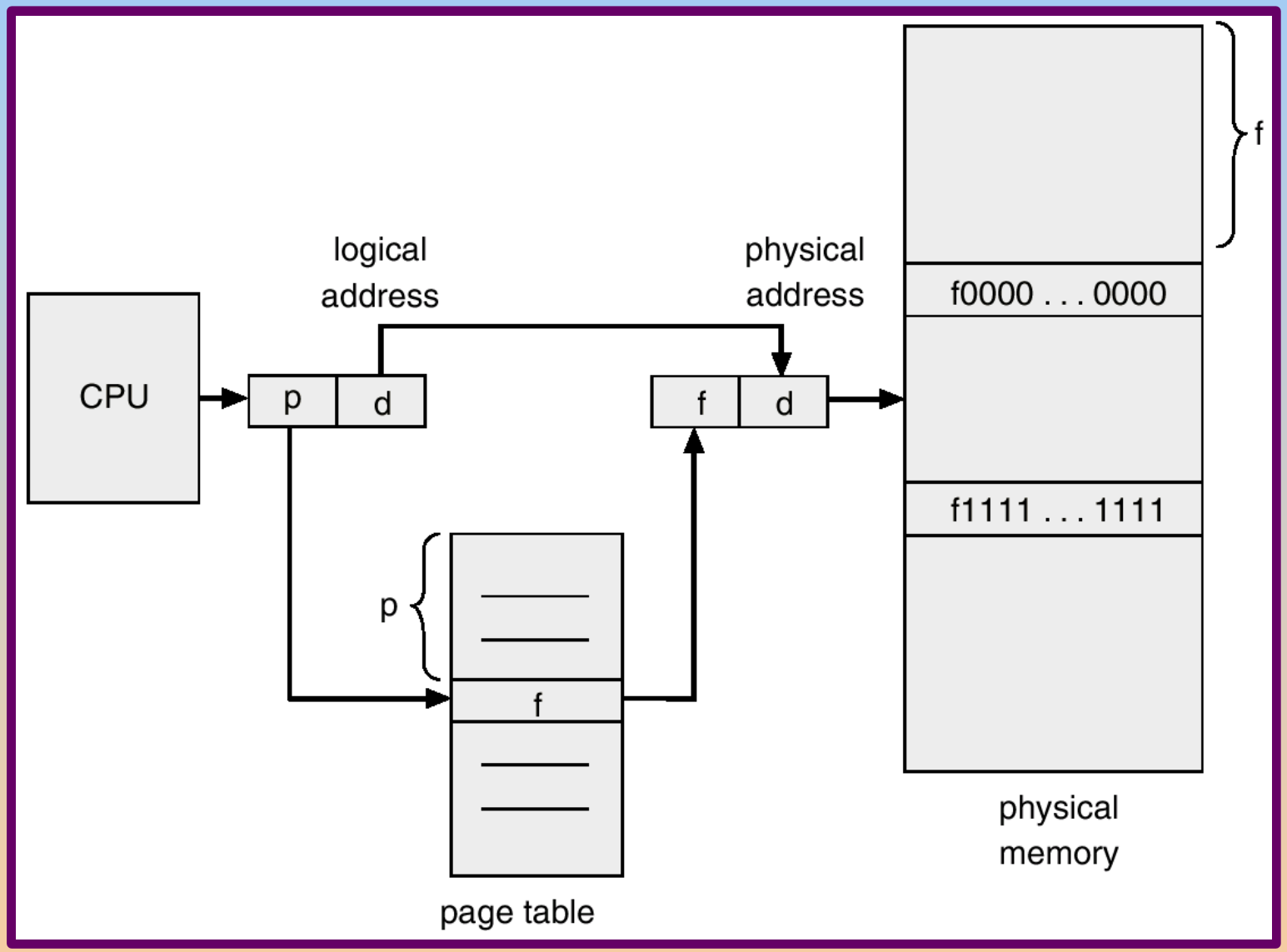
Paginação

- O espaço de endereçamento lógico de um processo pode ser não contíguo;
- Dividir a memória física em blocos de tamanho fixo chamados **quadros**(frames), normalmente potência de 2, entre 512 e 8192 bytes.
- Dividir a memória lógica em blocos de mesmo tamanho chamados **páginas**.
- Manter registro de todos os quadros livres.
- Para rodar um programa dividido em n páginas, o SO precisa encontrar n quadros livres e carregar o programa.
- Usar uma tabela de páginas para traduzir o endereço lógico para um endereço físico.
- Pode ocorrer fragmentação interna.

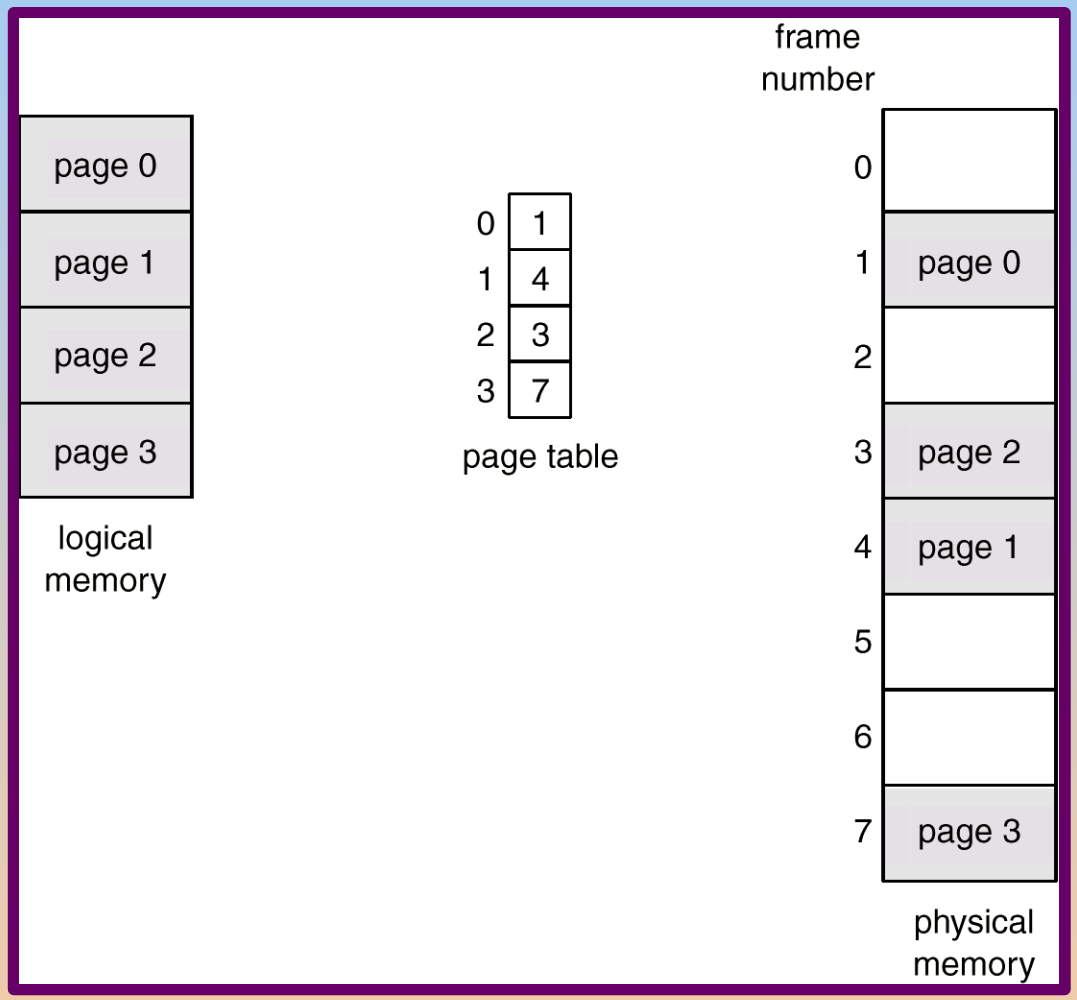
Esquema de tradução de endereço

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

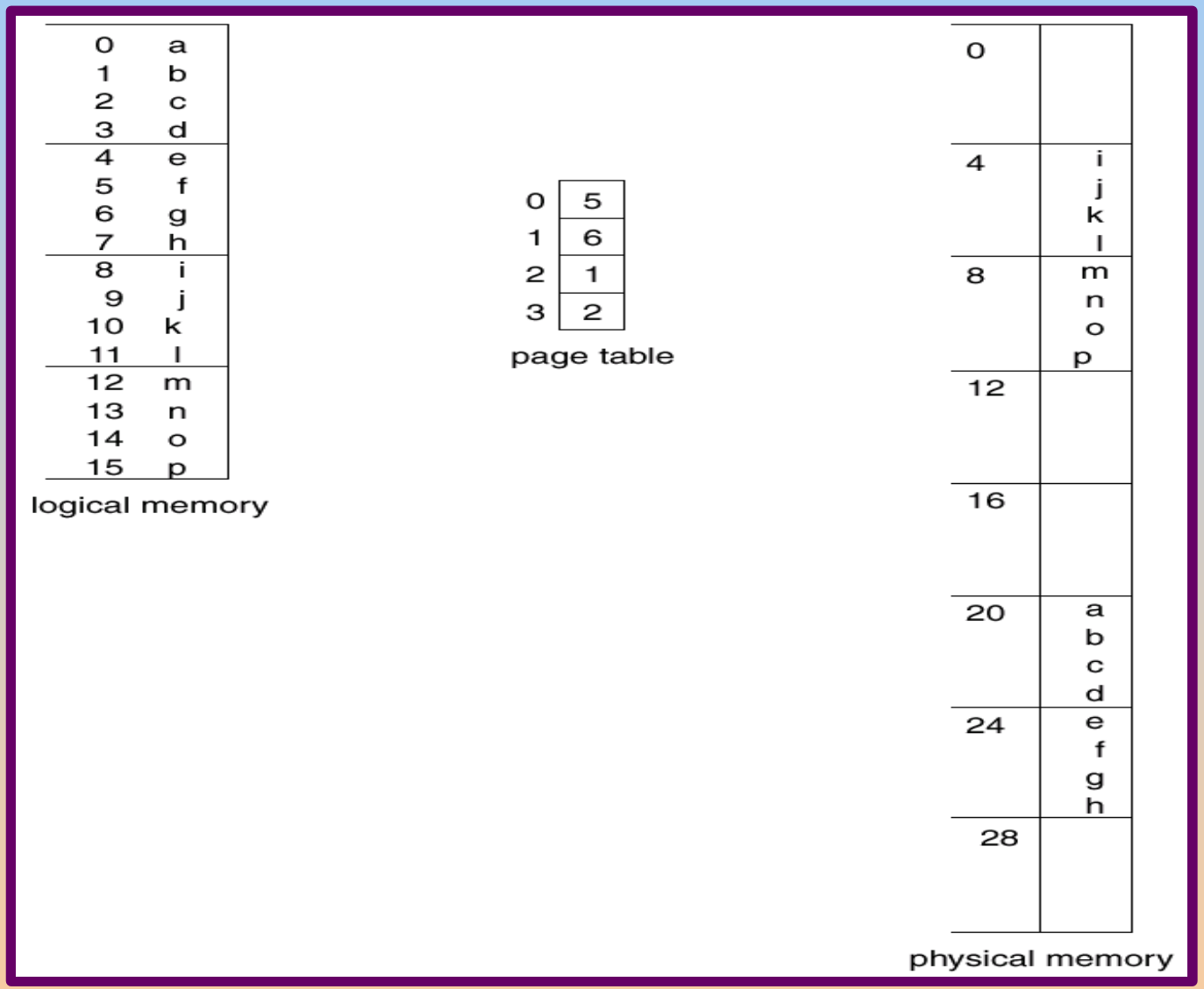
Arquitetura de tradução de endereço



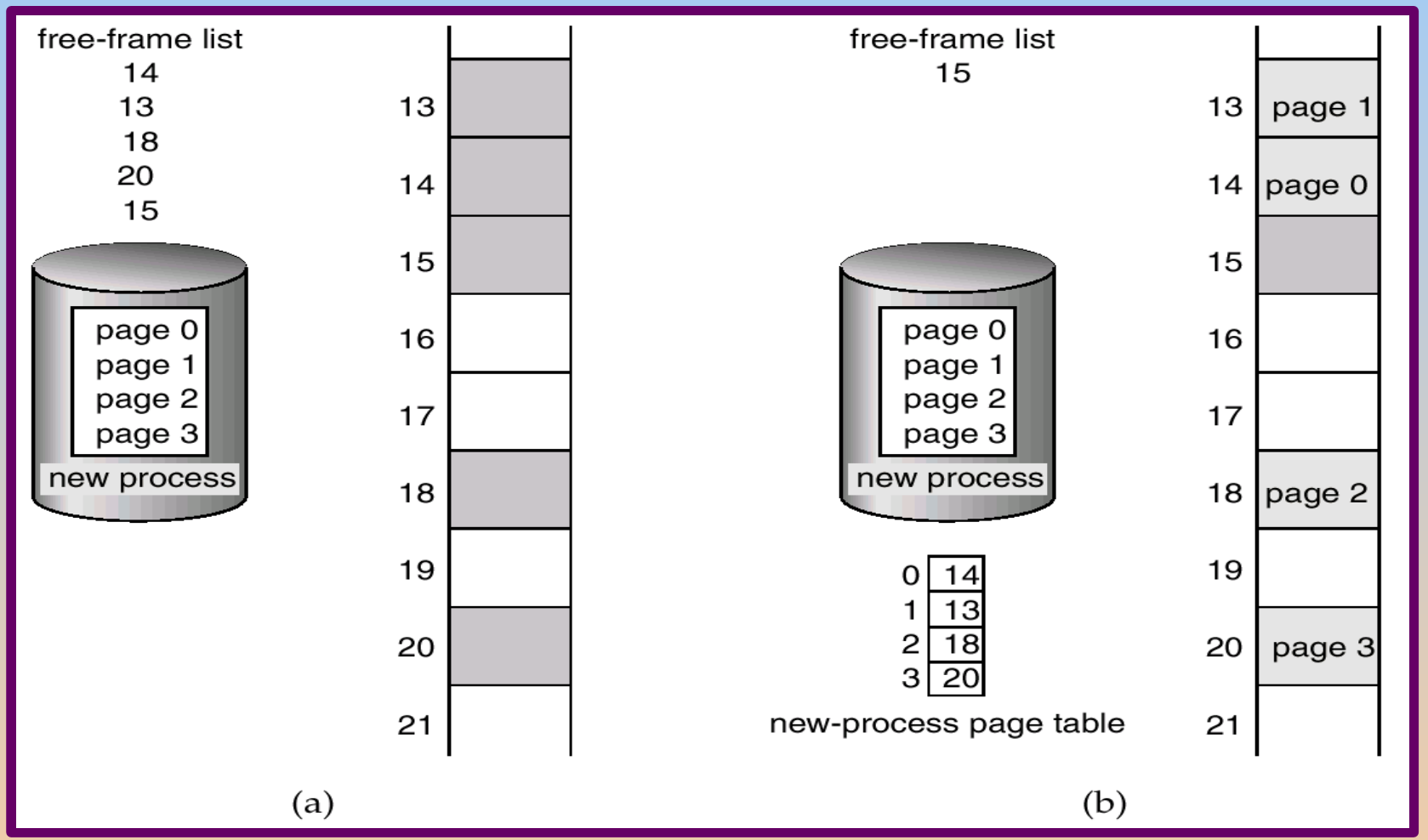
Exemplo de paginção



Exemplo de paginação



Quadros livres



Antes da alocação

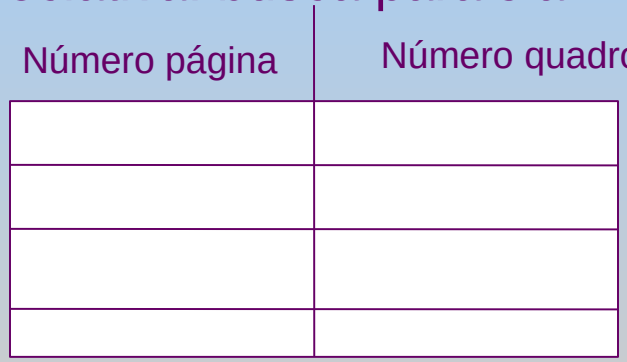
Depois de alocação

Implementação da tabela de páginas

- Tabela de páginas é mantida na memória principal
- *Registrador-base para a tabela de páginas (*PTBR) aponta para a tabela de páginas.
- *Registrador de tamanho da tabela de páginas* (PRLR) indica o tamanho da tabela de páginas.
- Neste esquema, cada acesso a dados/instrução requer dois acesso à memória. Um para a tabela de páginas e outro para dados/instrução.
- O problema dos dois acessos à memória podem ser resolvidos usando usando um hardware de cache especial de busca rápida chamado memória associativa ou registradores associativos(*translation look-aside buffers (TLBs)).*
- Um TLB armazena somente algumas entradas da tabela de páginas.

Memória associativa

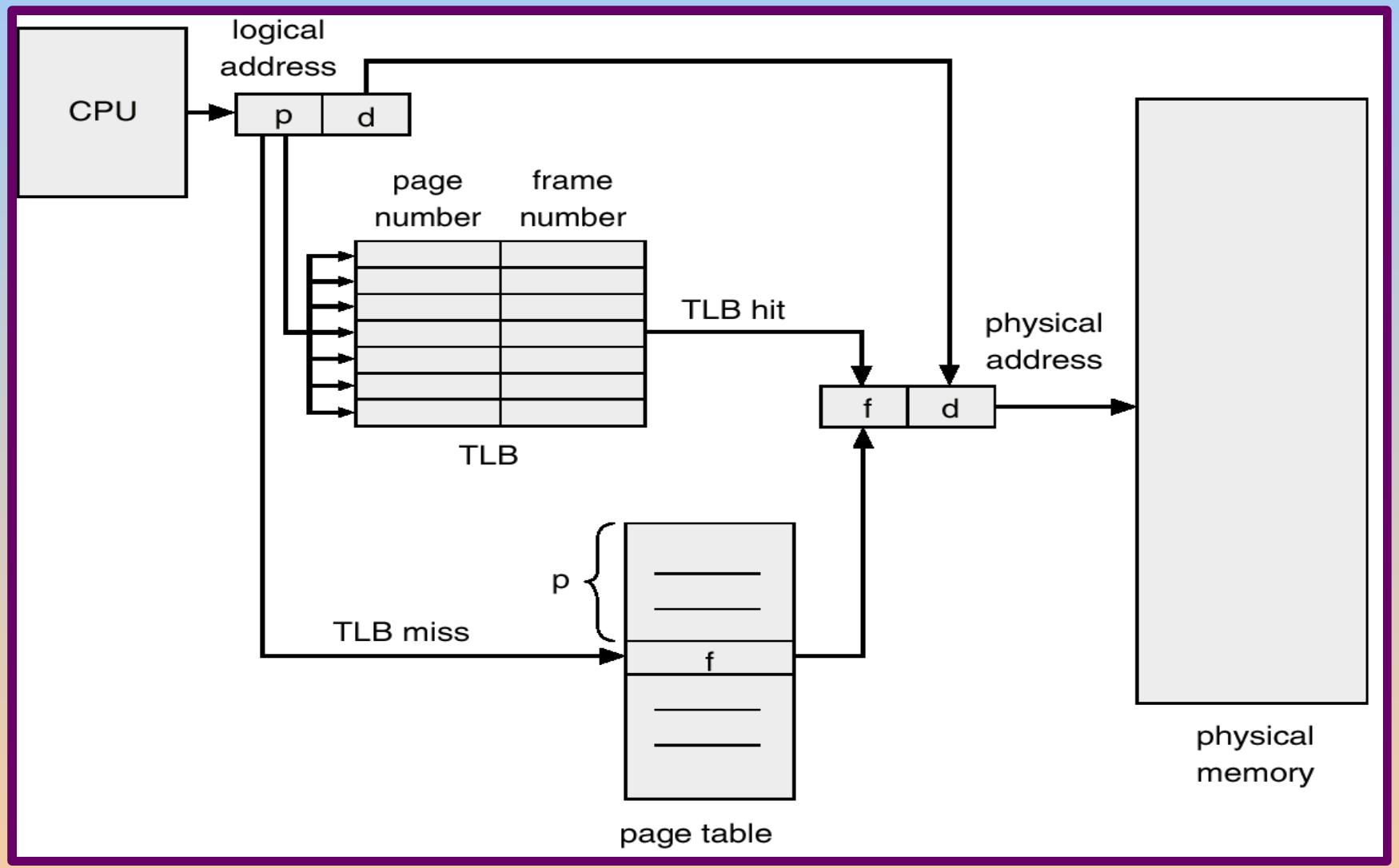
- Memória associativa: busca paralela



Tradução de endereço (A´, A´´)

- Se A’ está no registrador associativo, devolva o número do quadro A’’.
- Caso contrário, carregue o número da página na memória associativa, juntamente com o número do frame.

Hardaware de paginação com TLB

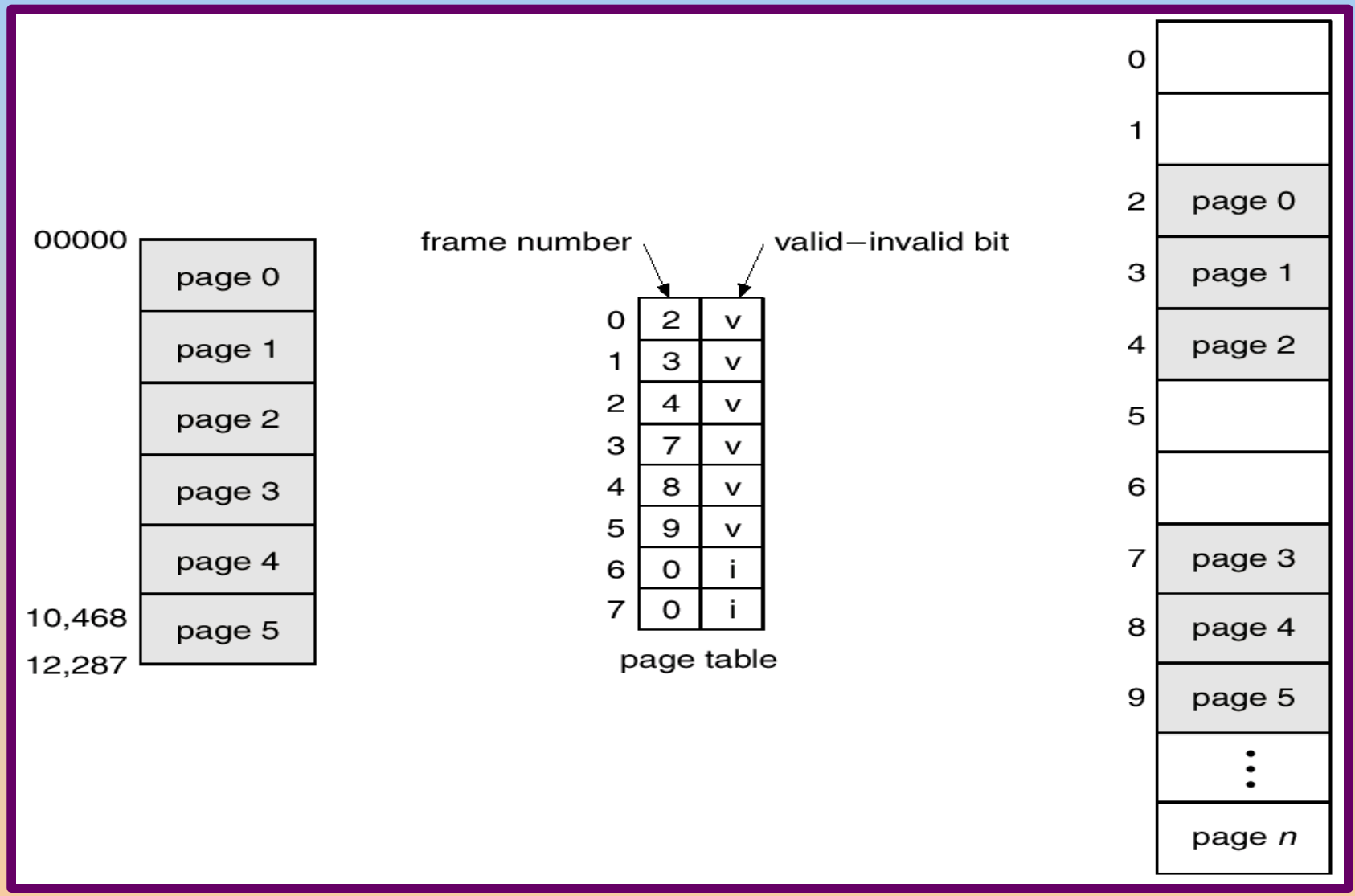


Proteção de memória

- Proteção de memória é implementada associando-se um bit de proteção com cada quadro.

- Um bit de válido-não válido é associado com cada entrada na tabela de páginas:
  - “válido” indica que a página associada está no espaço de endereçamento lógico do processo e é uma página legal.
  - “inválido” indica que a página não está no espaço de endereçamento lógico do processo.

Bits válido (v) ou inválido (i) Bit em uma tabela de páginas



Estrutura da tabela de páginas

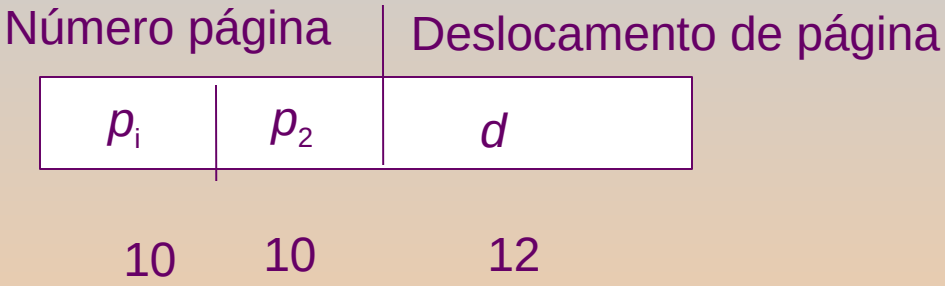
- Paginação hierárquica
- Tabelas de paginação com hashing
- Tabelas com paginação invertida

Tabelas de paginação hierárquica

- Quebra o espaço de endereámento lógico em várias tabelas de páginas.
- Um técnica simples é a paginação em dois níveis.

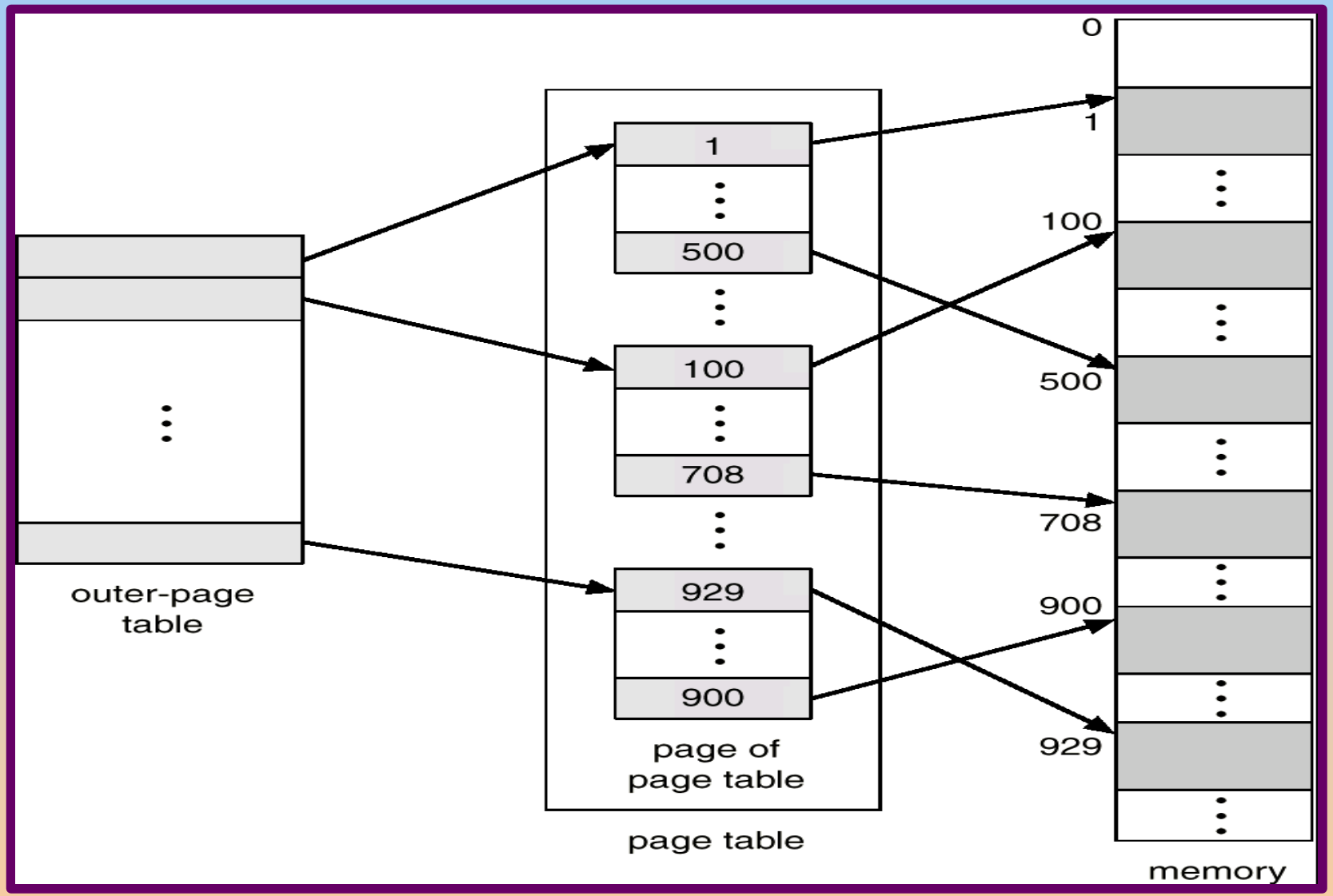
Exemplo de paginação em dois níveis

- Um endereço lógico (em máquinas de 32 bits com tamanho de página 4k) é dividido em:
  - Um número de página consistindo de 20 bits.
  - Um deslocamento na página de 12 bits.
- Como a tabela de páginas será paginada, o número da página é dividido em:
  - Um número de página com 10 bits.
  - Um deslocamento de página com 10 bits.
- Então, um endereço lógico é como:



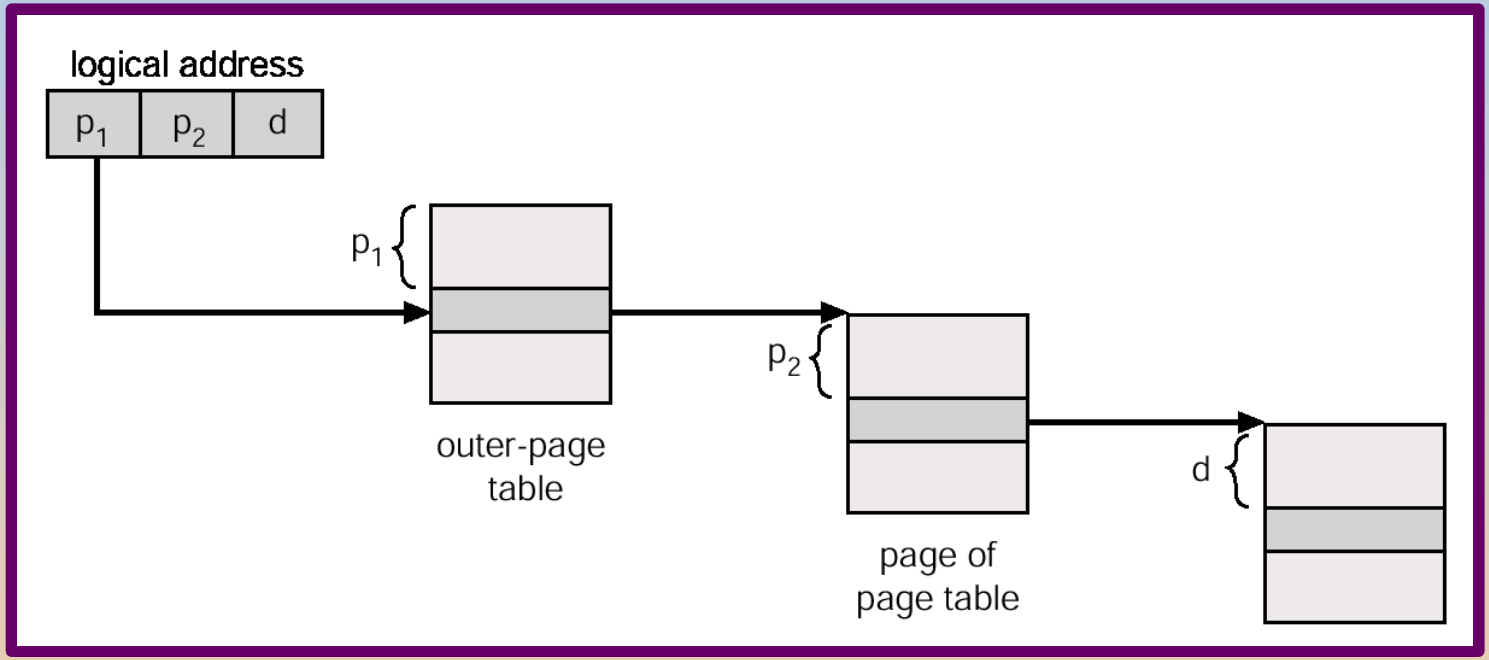
onde p_i é um índice numa tabela externa(outter-table), e p_2 é o deslocamento dentro da tabela interna.

Esquema de tabela de páginas em dois níveis



Esquema de tradução de endereço

- Esquema de tradução de ndereço com arquitetura de paginação 32-bits em dois níveis.



Tabelas de paginação com hashing

- Comum em espaços de endereçamento > 32 bits.
- Utiliza-se hashing com o número da página. Na entrada correspondente a esta chave existe uma lista ligada de páginas mapeados para a mesma posição.
- Efetua-se uma busca nesta lista de páginas. Encontrando-se a página desejada, o número do quadro é retornado.

Tabela de paginação com hashing

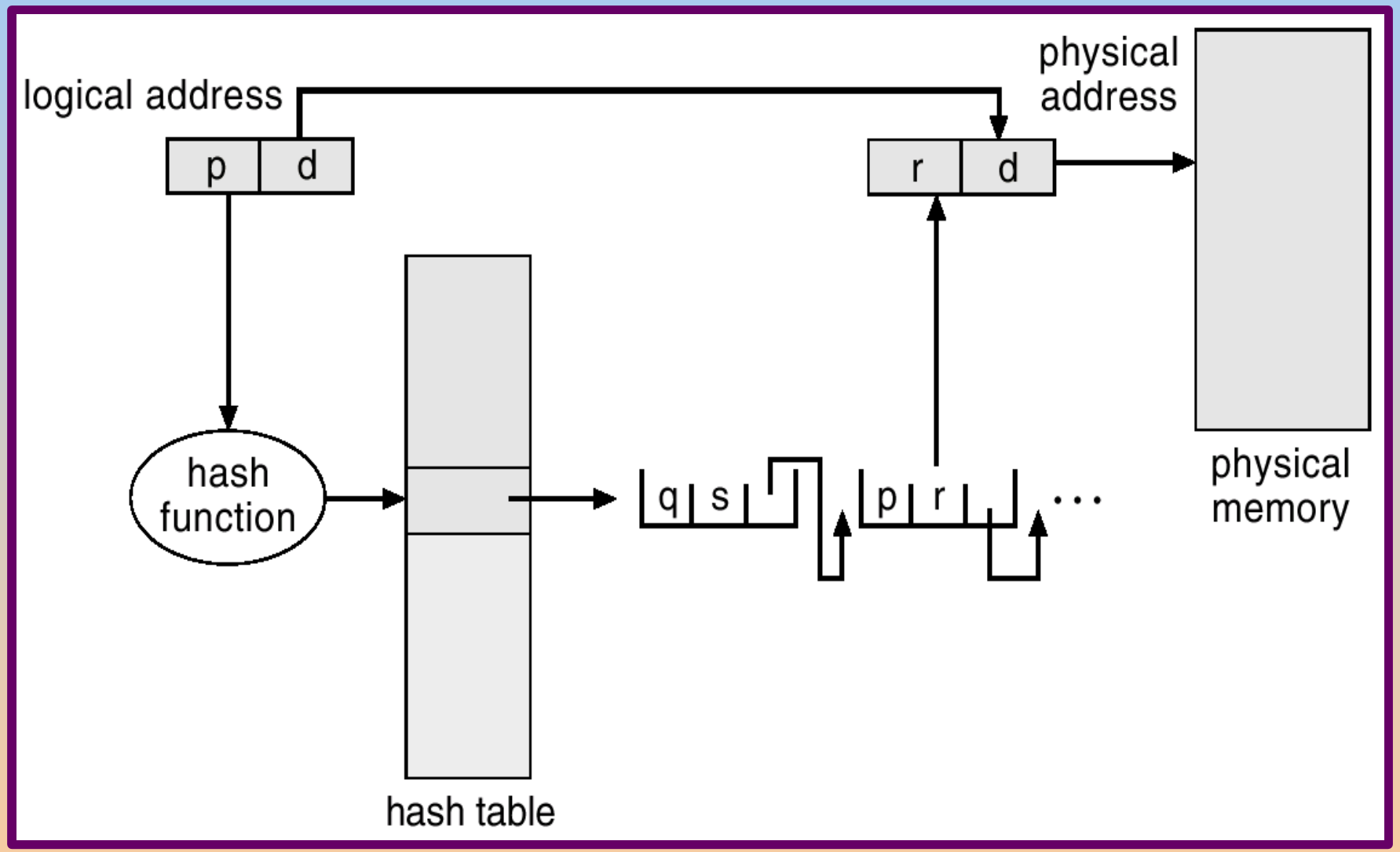
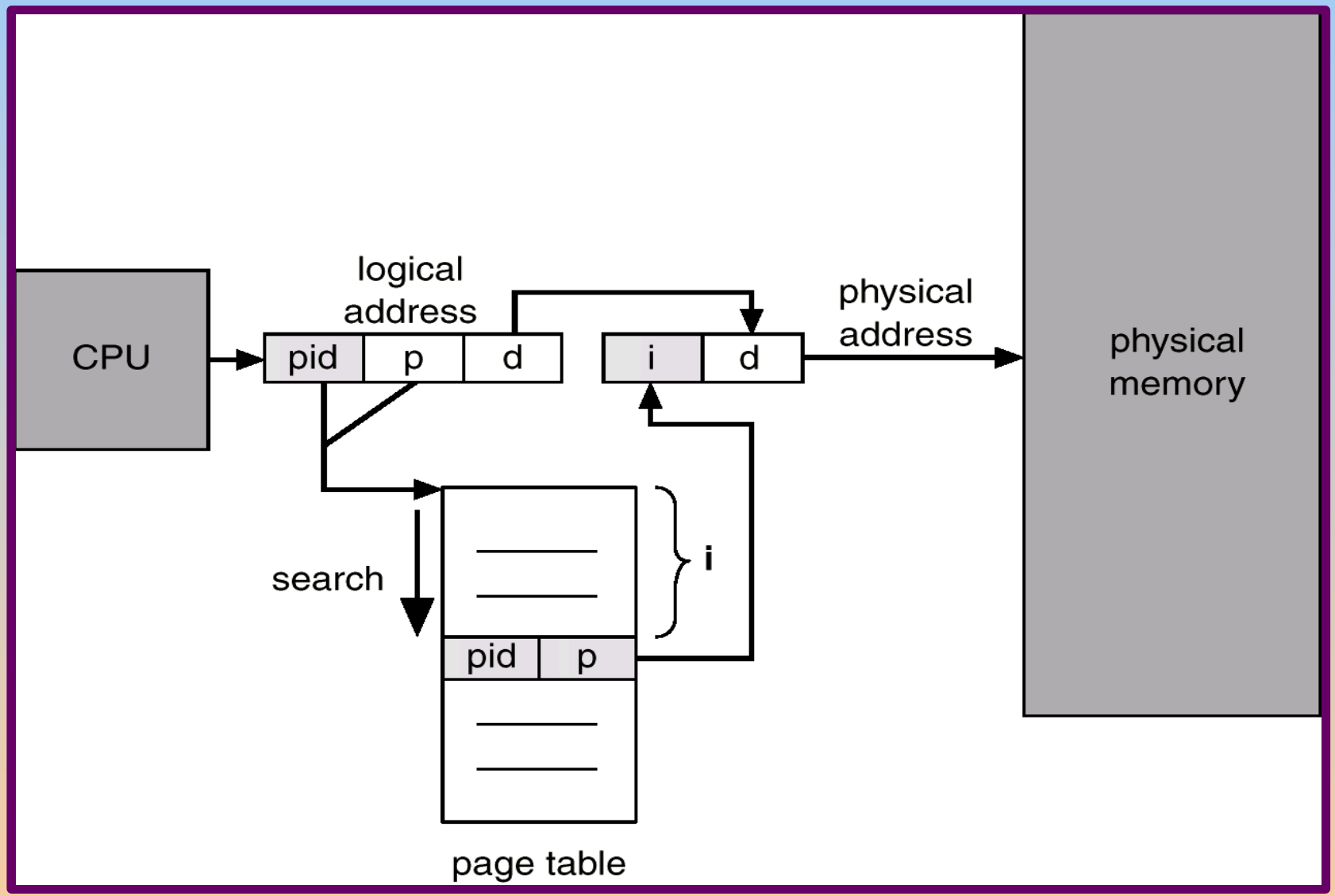


Tabela de páginas invertida

- Uma entrada para cada página real na memória.
- Entrada consiste de um endereço virtual da página armazenada numa localização da memória principal, com informação sobre o processo proprietário desta página.
- Diminiu o espaço de memória necessário para armazenar cada tabela de páginas, mas aumenta o tempo necessário para buscar a tabela quando uma referência de página ocorre.
- Pode utilizar uma tabela de hashing para limitar a busca a um elemento ou, no máximo, poucas entradas na tabela de páginas.

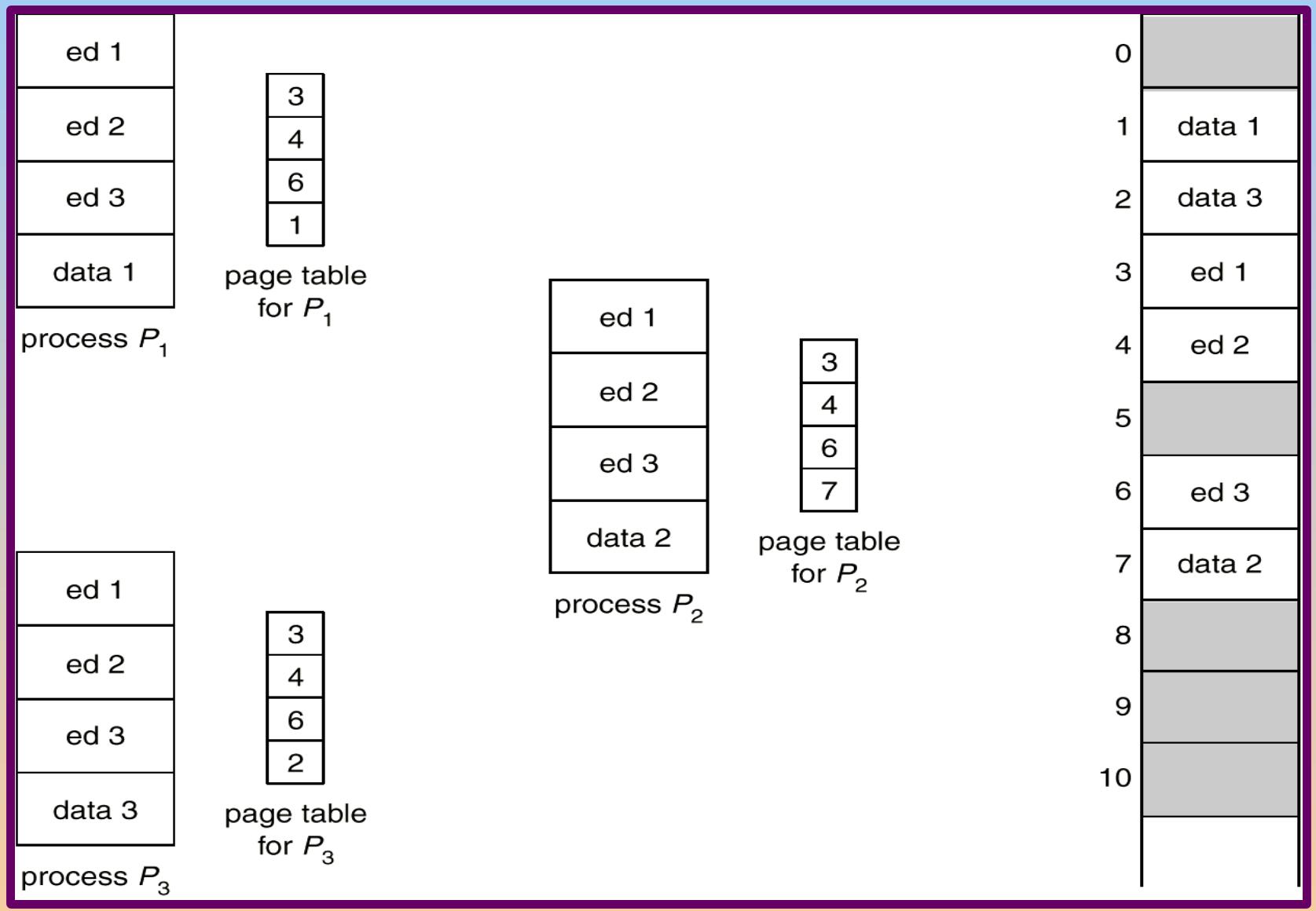
Arquitetura de tabela de páginas invertidas



Páginas compartilhadas

- Código compartilhado
  - Uma cópia de código de somente leitura compartilhado entre processos (código reentrante). Exemplo: editores de texto, compiladores, sistemas de janela.
  - Código compartilhado precisa aparecer na mesma localização no espaço de endereámento lógico de todos os processos.
- Código e dados privados
  - Cada processo mantém uma cópia separada de código e dados
  - As páginas para código privado e dados podem aparecer em qualquer lugar no espaço de endereçamento lógico.

Exemplo de páginas compartilhadas



Segmentação

- Esquema de gerenciamento de memória que suporta visão do usuário da memória.
- Um programa é uma coleção de segmentos. Um segmento é unidade lógica, tais como:

  Programa principal,

  procedimento,

  função,

  método,

  objeto,

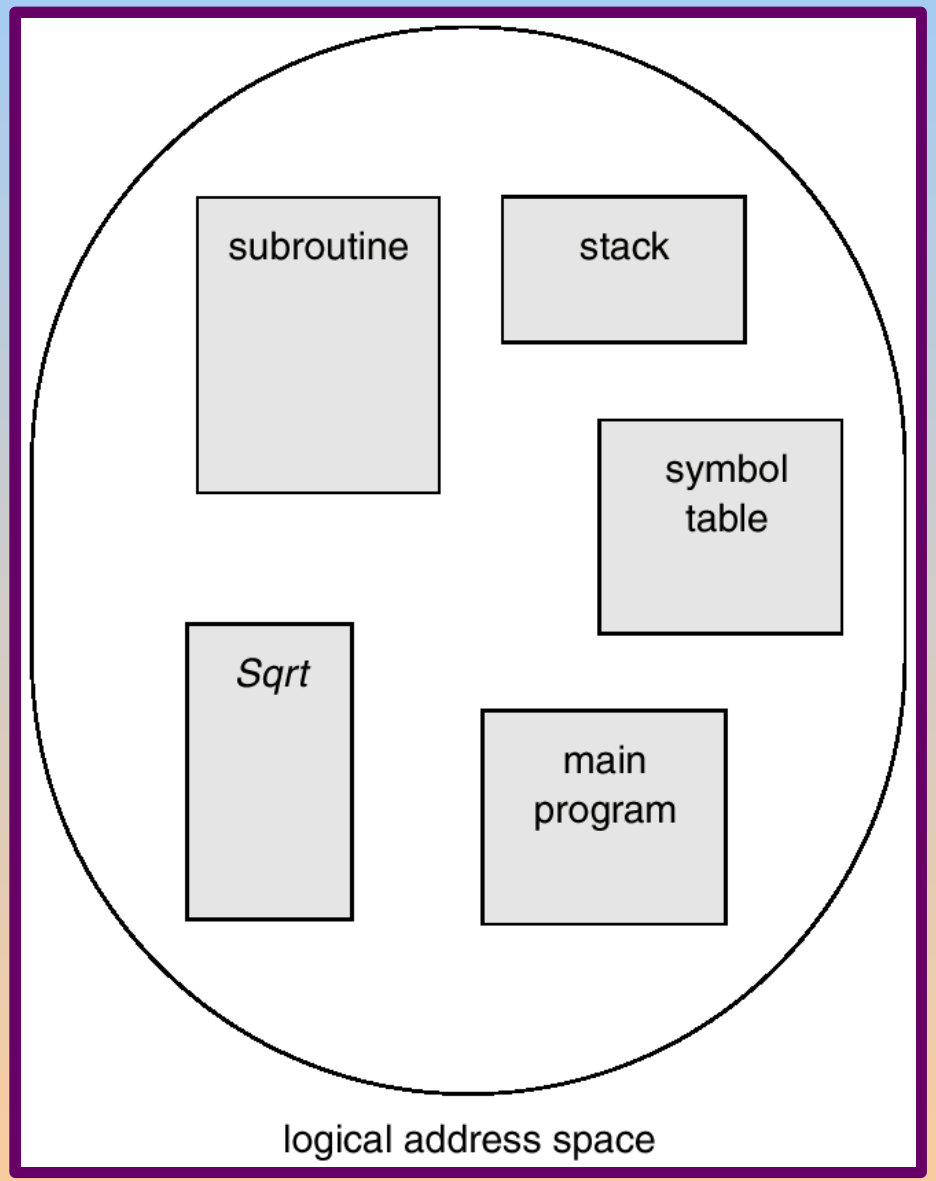
  variáveis locais, variáveis globais,

  blocos comuns,

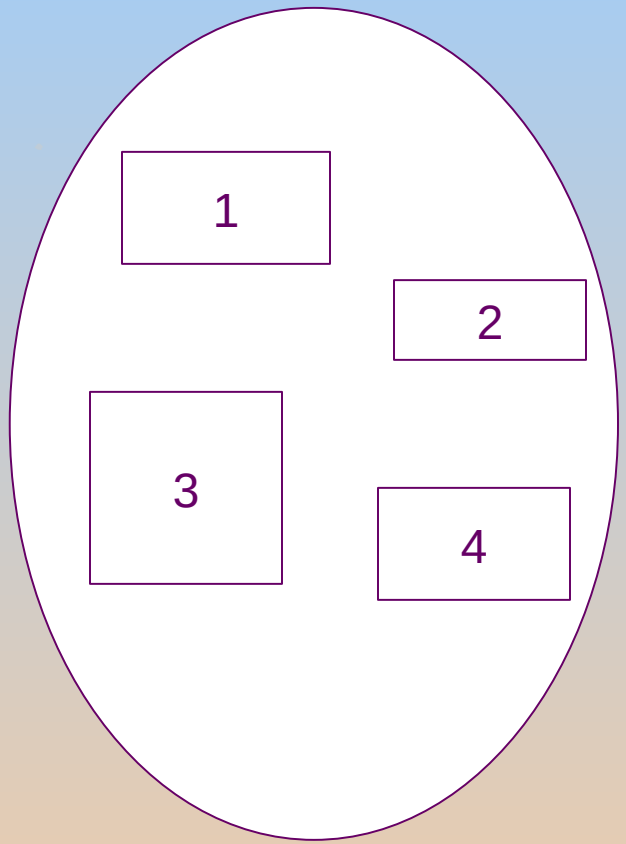
  pilha,

  tabela de símbolos, vetores

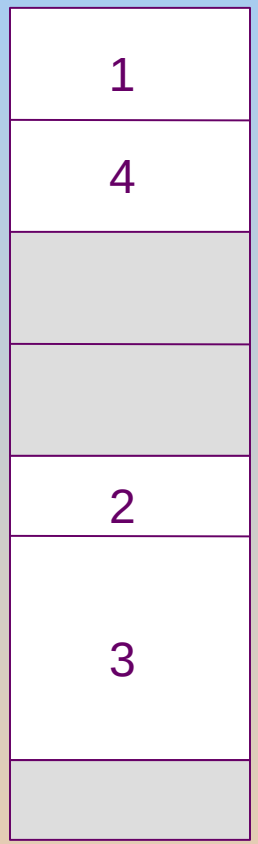
Visão de usuário de um programa



Visão lógica da segmentação



Espaço do usuário



Espaço de memória física

Arquitetura de segmentação

- Endereço lógico consiste de dois campos:

  <número-segmento, deslocamento>,
- *Tabela de segmentos* – mapeia endereços lógicos em endereços físicos; cada entrada da tabela contém
  - base – contém o endereço físico inicial onde os segmentos estão na memória.
  - *limite* – especifica o tamanho do segmento
- *Segment-table base register (STBR)* aponta para a localização da tabela de segmentos em memória.
- *Segment-table length register (STLR)* indica o número de segmentos usados por um programa

  número de segmento *s* é válido se *s* < STLR.

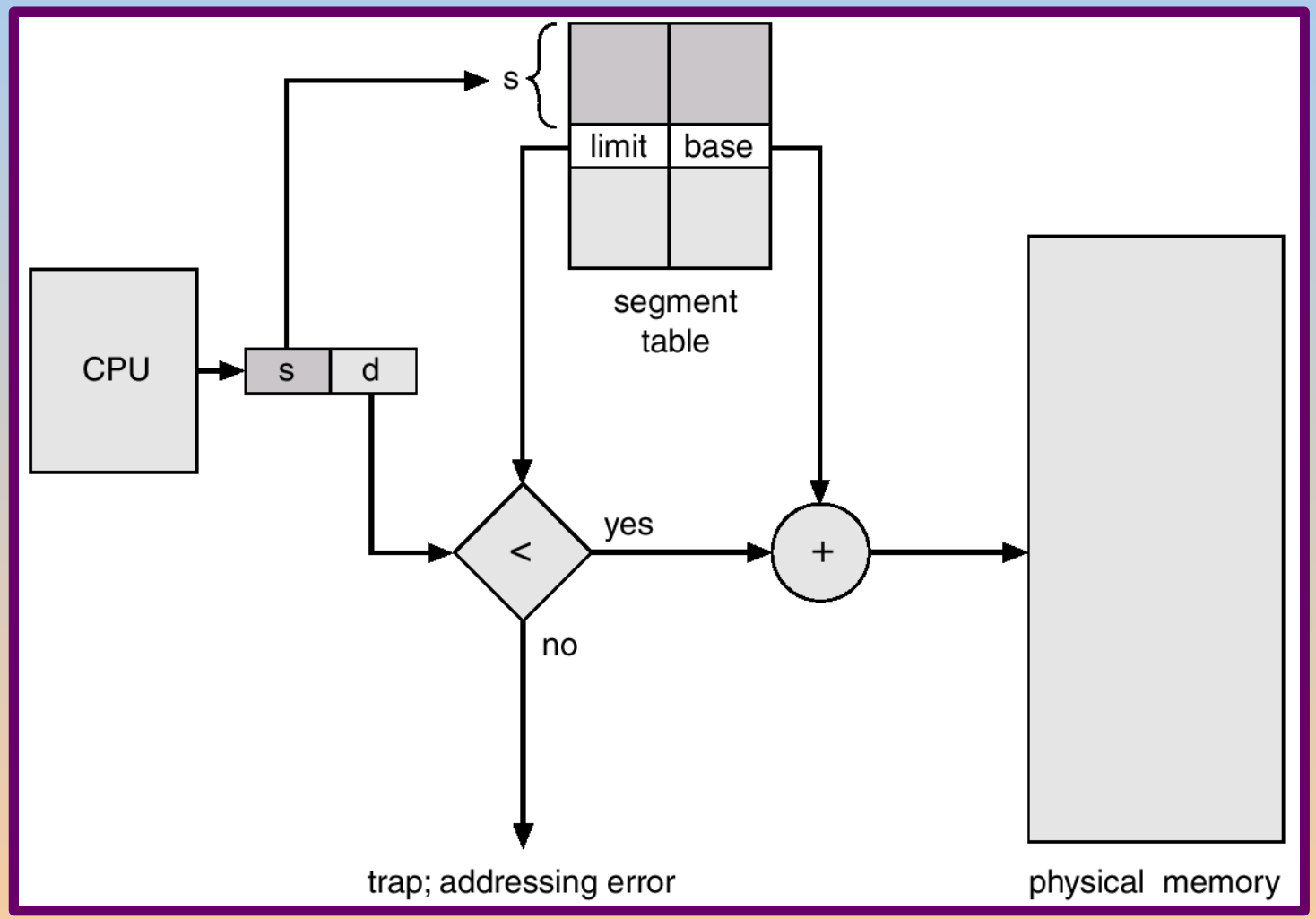
Arquitetura de segmentação (Cont.)

- Relocação.
  - dinâmica
  - Pela tabela de segmentos
- Compartilhamento.
  - Segmentos compartilhados
  - Mesmo número de segmento
- Alocação.
  - first fit/best fit
  - Fragmentação externa

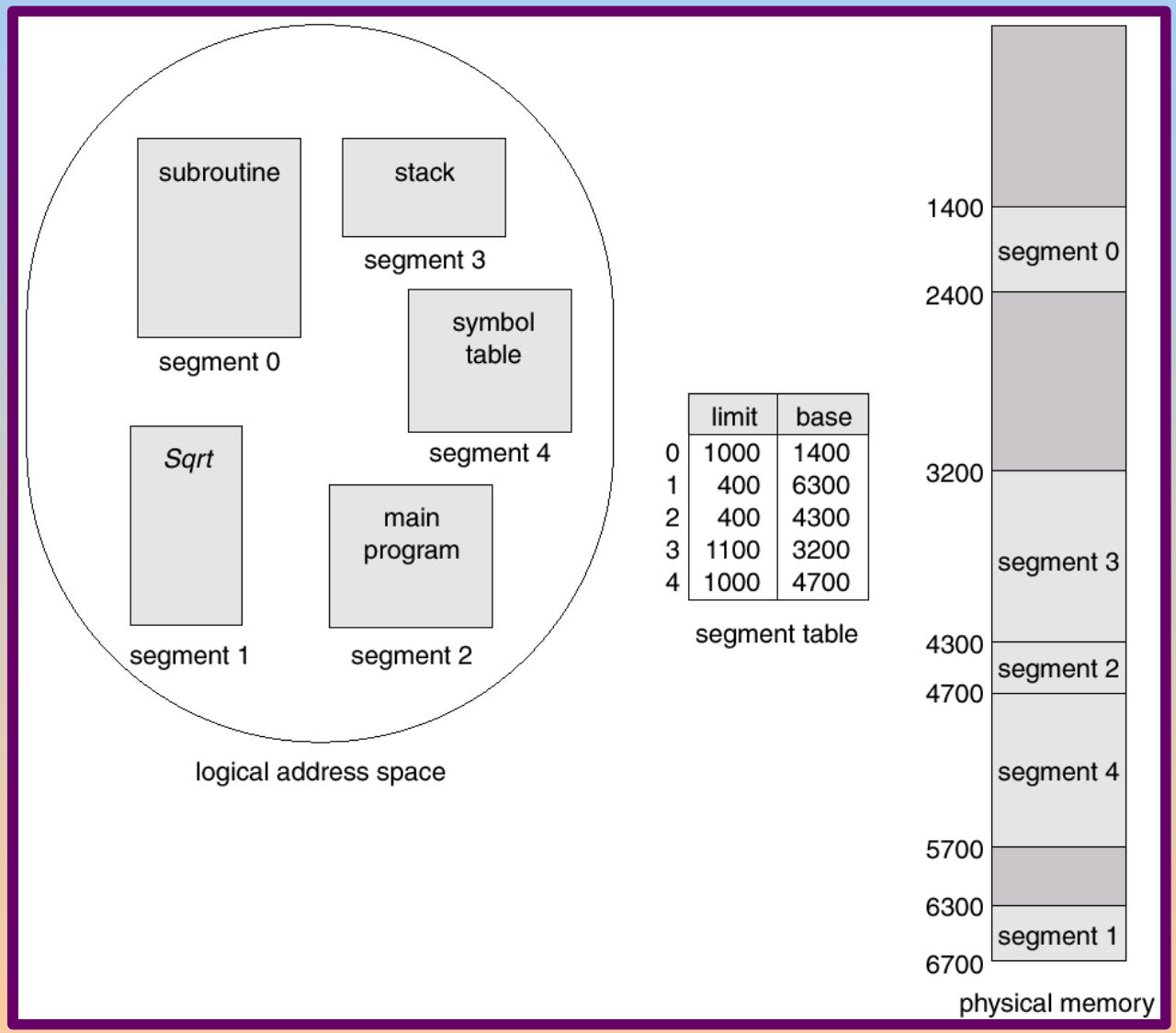
Arquitetura de segmentação (Cont.)

- Proteção. Cada entrada da tabela de segmentos contém:
  - Bit da validação = 0 $\Rightarrow$ segmento ilegal
  - Privilégios de leitura/escrita/execução
- Possui bits de proteção associados com segmentos; compartilhamento de código ocorre no nível de segmento
- Como os segmentos variam de tamanho, alocação de memória é uma problema de alocação dinâmica de recursos.

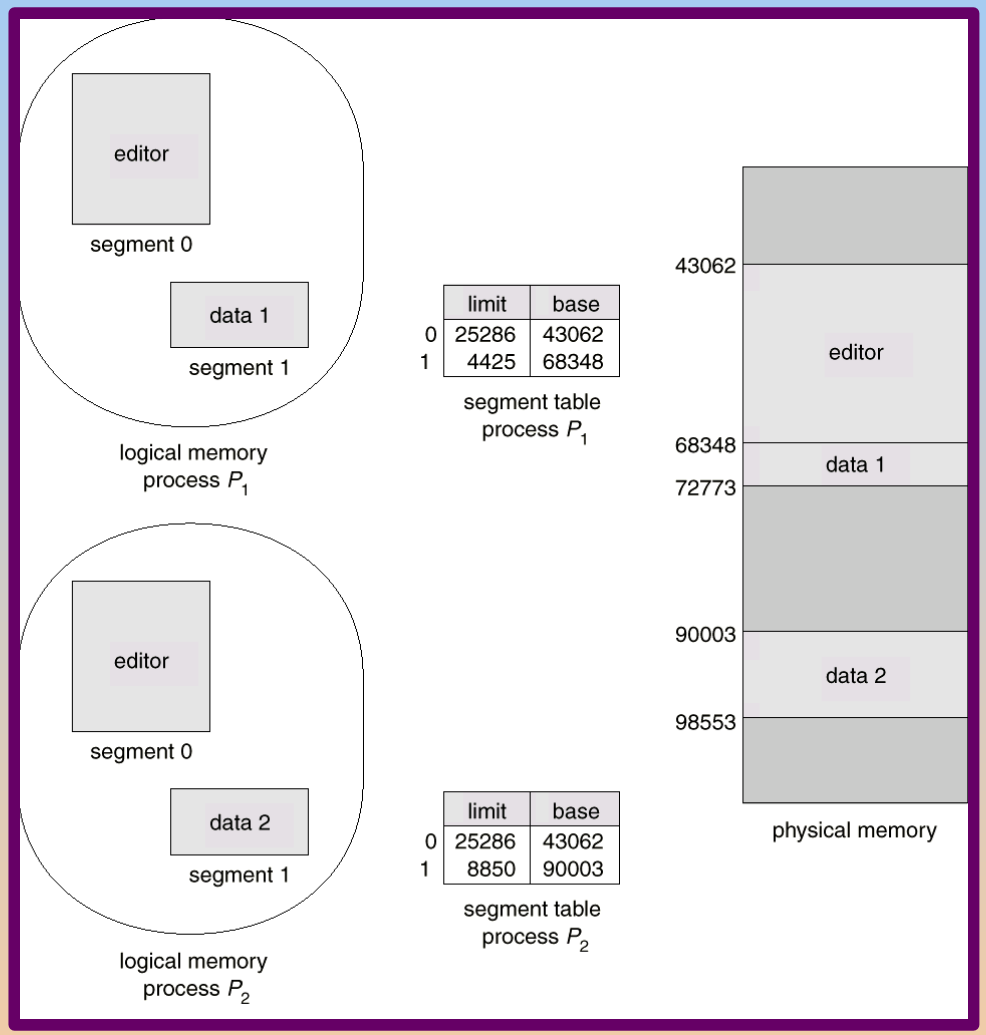
Hardware de segmentação



Exemplo de segmentação



Compartilhamento de segmentos

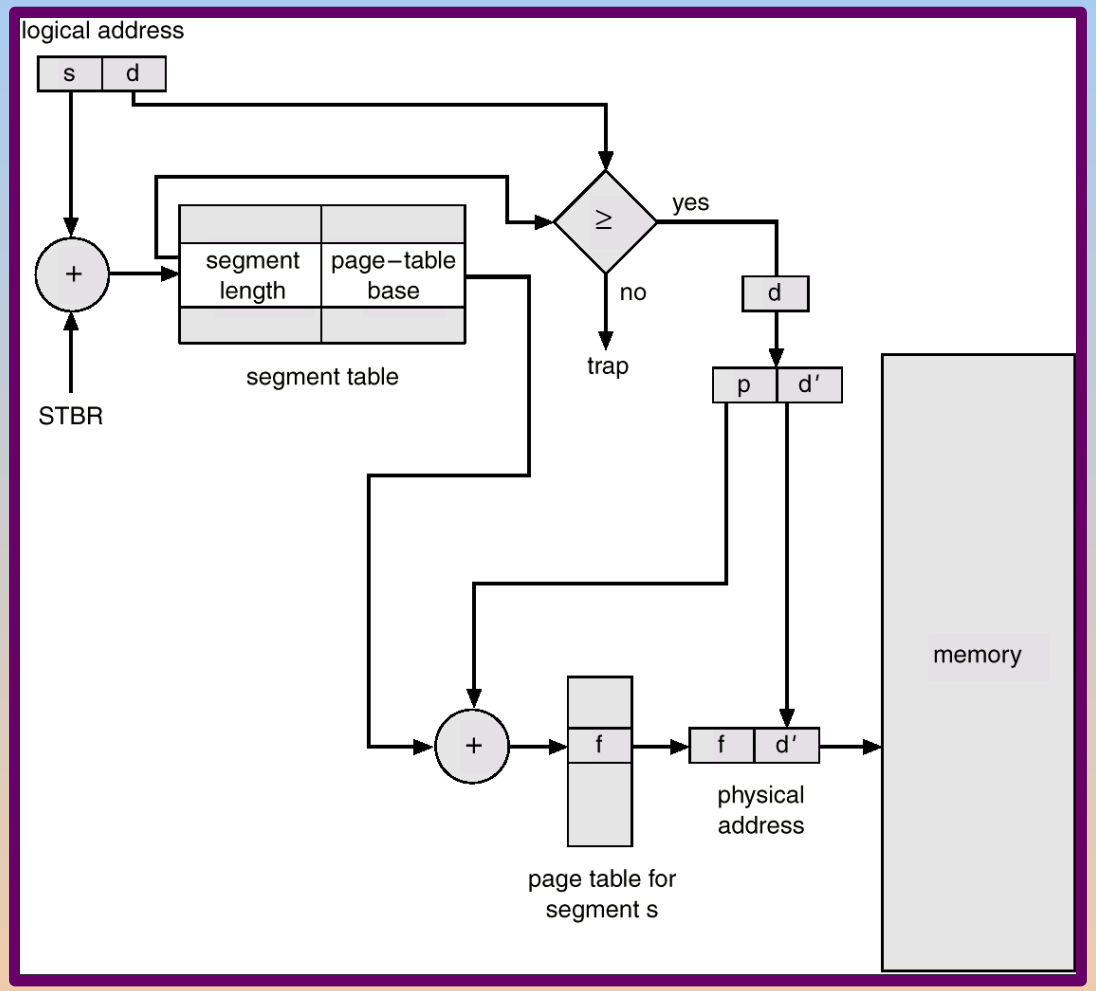


Segmentação com paginação – MULTICS

- O sistema MULTICS resolveu os problemas de fragmentação externa utilizando paginação de segmentos.

- Solução difere da segmentação pura no sentido em que a entrada da tabela de segmentos não contém mais o endereço do segmento, mas um endereço-base para a tabela de páginas deste segmento.

Esquema de tradução de endereço do MULTICS



Segmentação com paginação – Intel 386

- O processador Intel 386 usava segmentação com paginação para gerenciamento de memória com um esquema de paginação em dois níveis.

Esquema de tradução de endereço Intel 386