FACULDADE DE TECNOLOGIA

SISTEMAS OPERACIONAIS

LAURO L. A. WHATELY

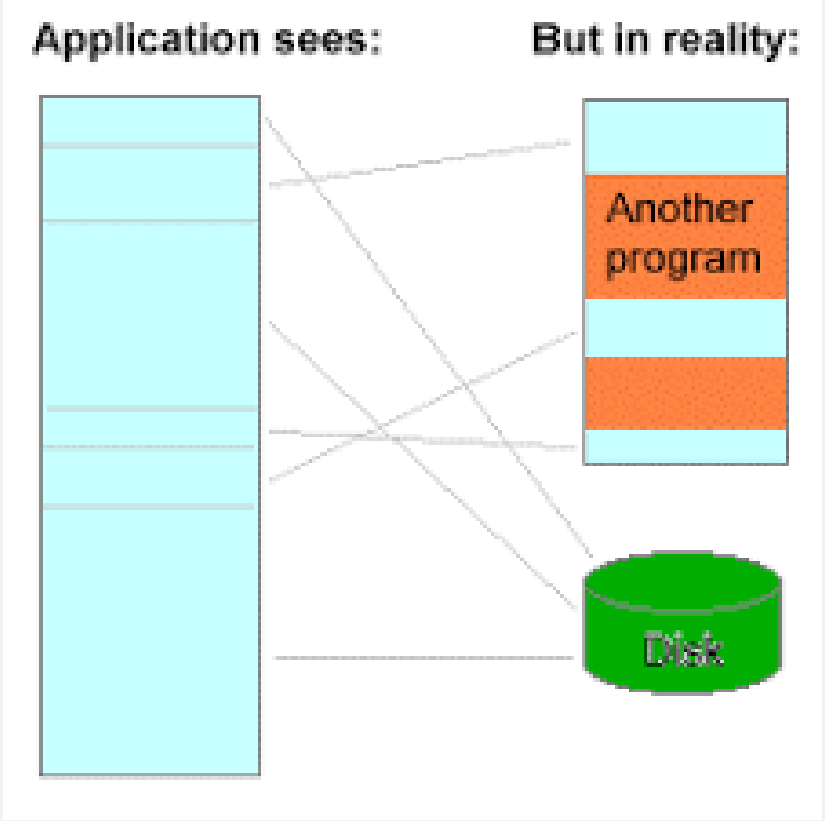
GERENCIAMENTO DE MEMÓRIA

OBJETIVOS

- Fornecer uma descrição detalhada das várias formas de organizar a memória do computador.
- Discutir várias técnicas de gerenciamento de memória, incluindo paginação e segmentação.
- Descrever como o sistema operacional cria abstração de memória virtual.

GERÊNCIA DE MEMÓRIA

- Fundamentos
- Troca de Processos (*Swapping*)
- Alocação Contígua
- Paginação
- Estrutura da Tabela de Páginas
- Segmentação



FUNDAMENTOS

- Programa deve ser trazido (do disco) para a memória e colocada dentro de um processo para ser executado.
- **Memória principal e registradores** são os únicos meios de armazenamento que a CPU acessa diretamente.
- **Cache** fica entre a memória e os registradores da CPU para diminuir o tempo de acesso.
- Proteção de memória é necessária para garantir a operação correta.

PROBLEMA

Em um ambiente multiprogramado, é necessário:

- Subdividir a memória para acomodar múltiplos processos;
- Processos, na memória, em boa parte do tempo estarão esperando por E/S:
  - processador sub-utilizado;
- Deve-se alocar memória de forma eficiente ao maior número de processos:
  - **Gerenciador de Memória.**

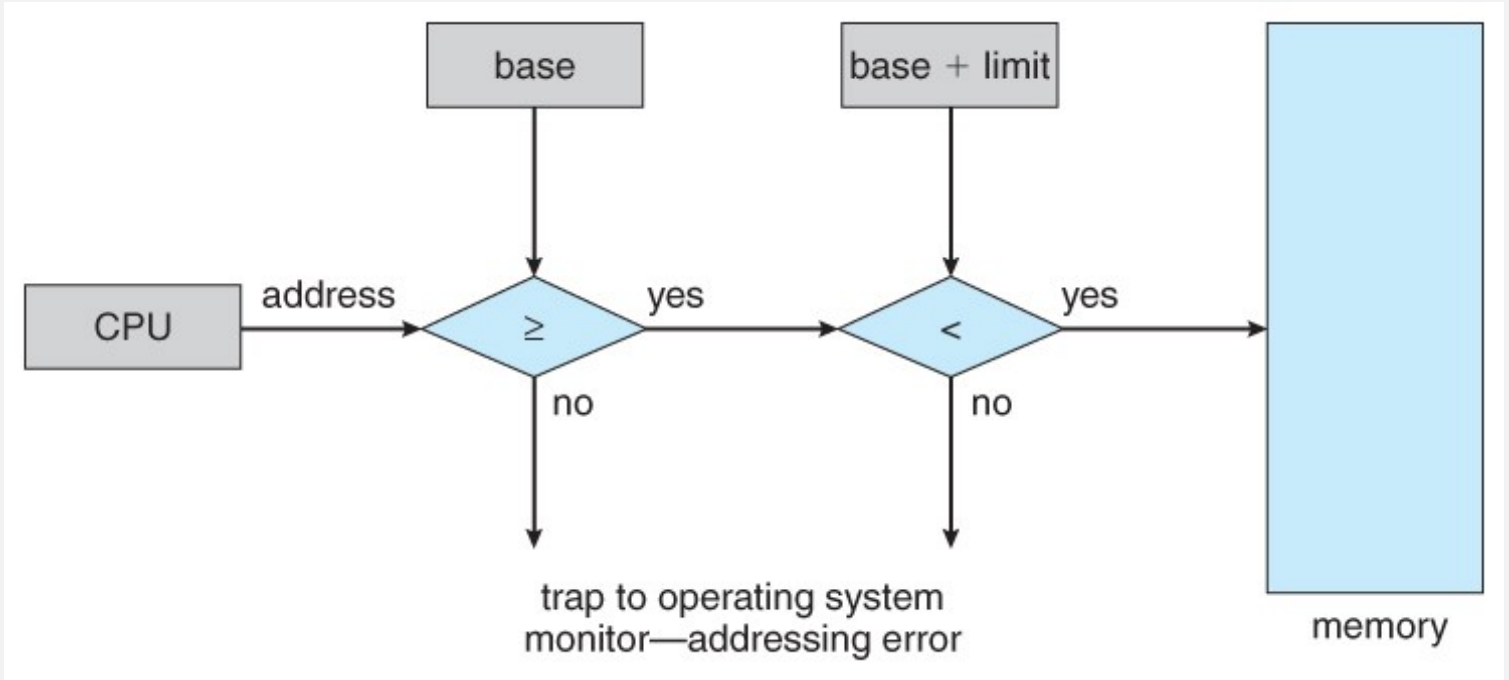
ESPAÇO DE ENDEREÇOS

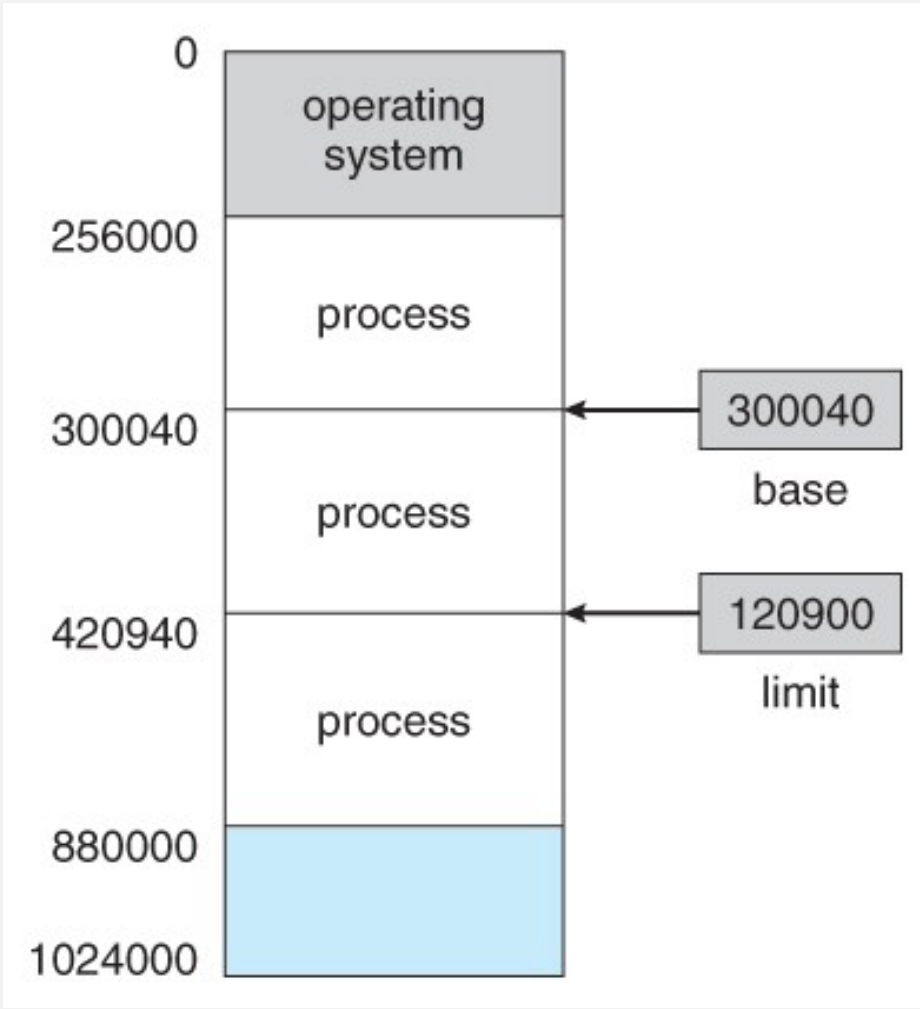
- Memória principal é organizada como um conjunto contínuo de células de 8 bits (byte ou octeto).
- A referência é feita por endereços físicos para bytes ou palavras de 4 ou 8 bytes.
- Em um ambiente multiprogramado, múltiplo programas podem residir na memória ao mesmo tempo.
- O programa não sabe a priori o local na memória que vai ser armazenado.
- **endereço lógico** - especifica uma posição genérica, relativa para o programa.
- **endereço físico** – um endereço real na memória principal.
- O processador gera um endereço lógico que precisa ser convertido para o endereço físico.

ENDEREÇO LÓGICO X FÍSICO

- O valor no registrador *base* é adicionado a cada endereço gerado pelo processo do usuário no momento que é enviado para a memória. O endereço deve estar dentro do limite dado no registrador *limite*.

- O programa do usuário lida com endereços *lógicos*; ele nunca trata os endereços físicos *reais*





ALOCAÇÃO CONTÍGUA

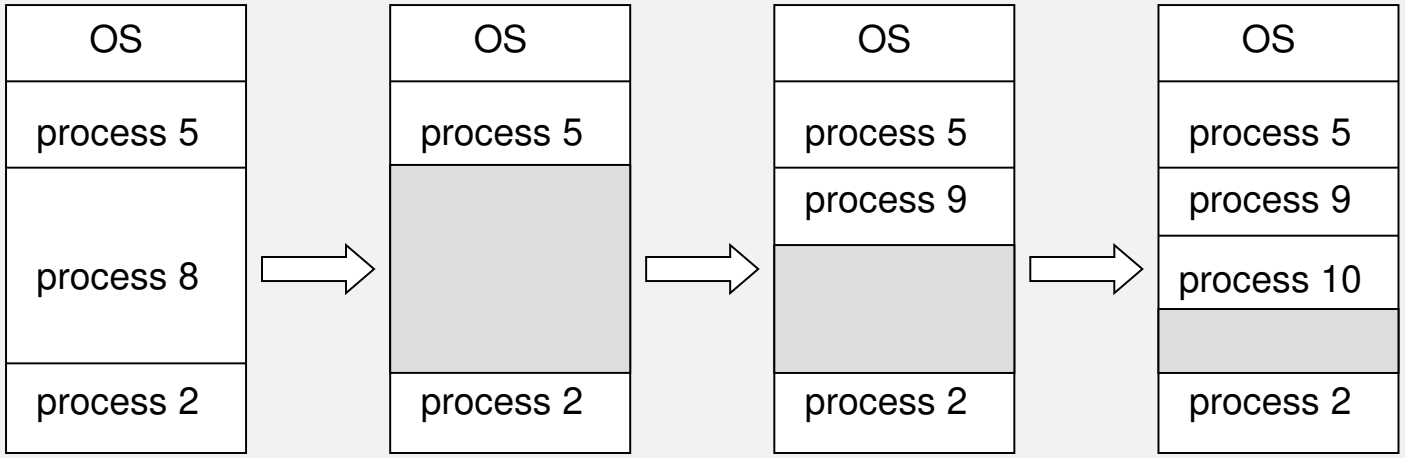
- A **memória principal** é normalmente divida em duas partes:
  - Parte residente do kernel, normalmente mantida na **parte baixa** da memória com o vetor de interrupções.
  - Processos do usuário mantidos na **parte alta** da memória.

- **Registradores de relocação** são usados para proteger processos dos usuários uns dos outros, e de alterar os códigos e dados do sistema operacional.
  - Registrador *base* contém o valor do menor endereço físico;
  - Registrador *limite* contém o tamanho do intervalo dos endereços lógicos – cada endereço lógico deve ser menor que o registrador limite.
  - MMU mapeia o endereço lógico dinamicamente.
  - Apenas o kernel pode modificar estes registradores.

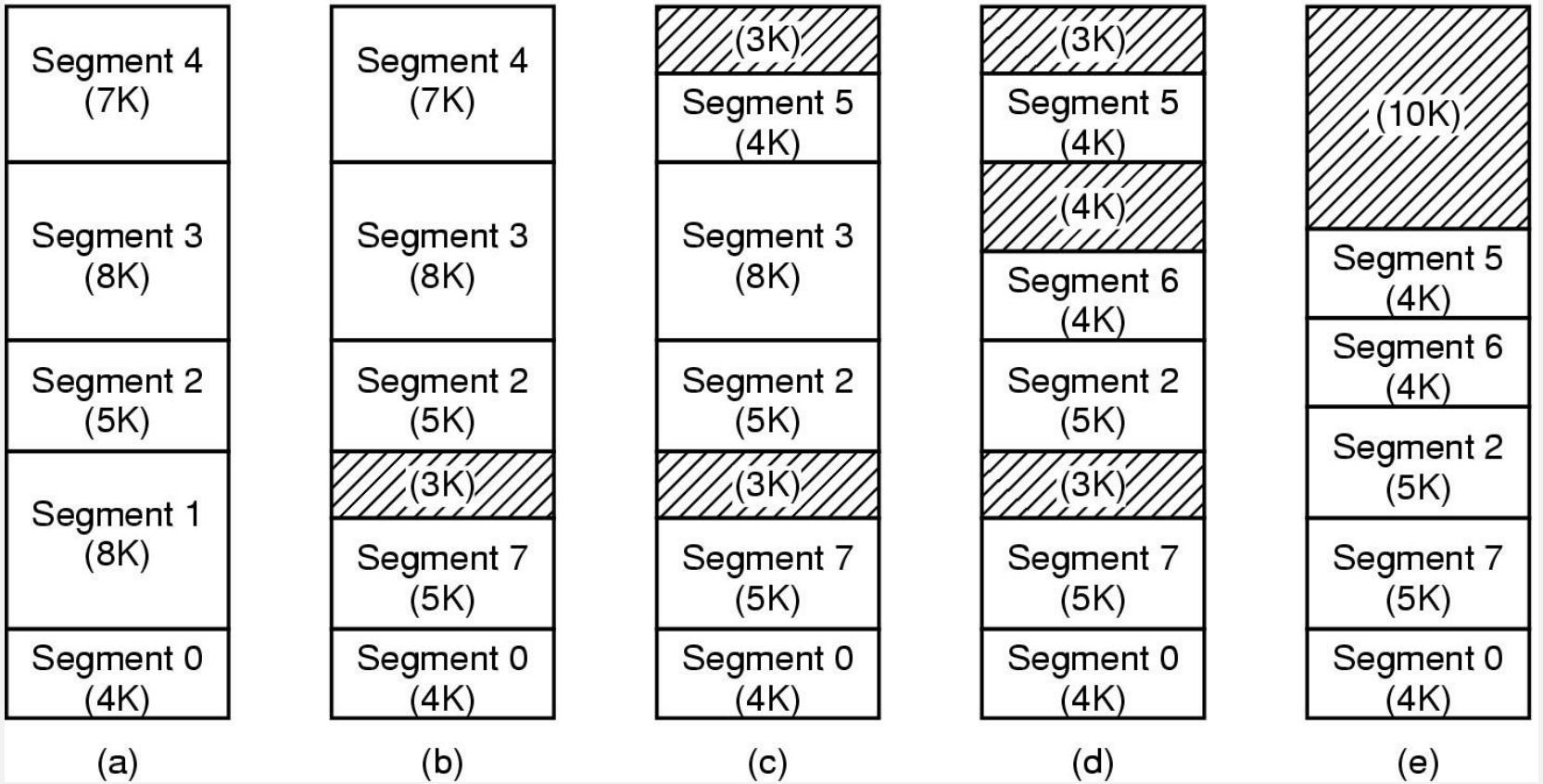
ALOCAÇÃO CONTÍGUA

Alocação com Diversas Partições

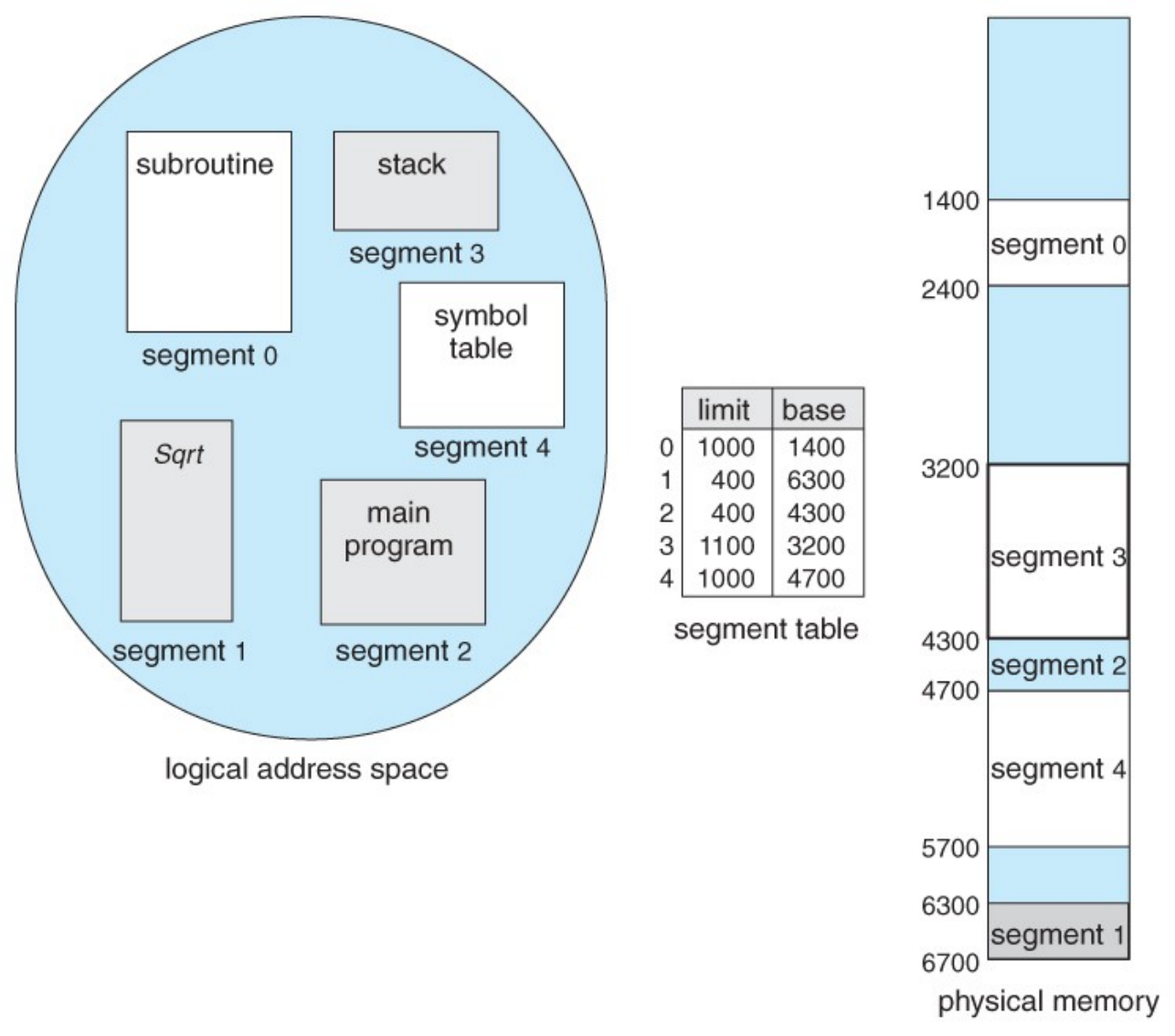
- *Bloco Livre (Hole)* – bloco de memória disponível; blocos de vários tamanhos são espalhados pela memória.
- Quando um processo chega, é alocada memória de um bloco livre grande o suficiente para acomodá-lo.
- Sistema Operacional mantém informações sobre:
  a) partições alocadas b) partições livres (*holes*)



SEGMENTAÇÃO



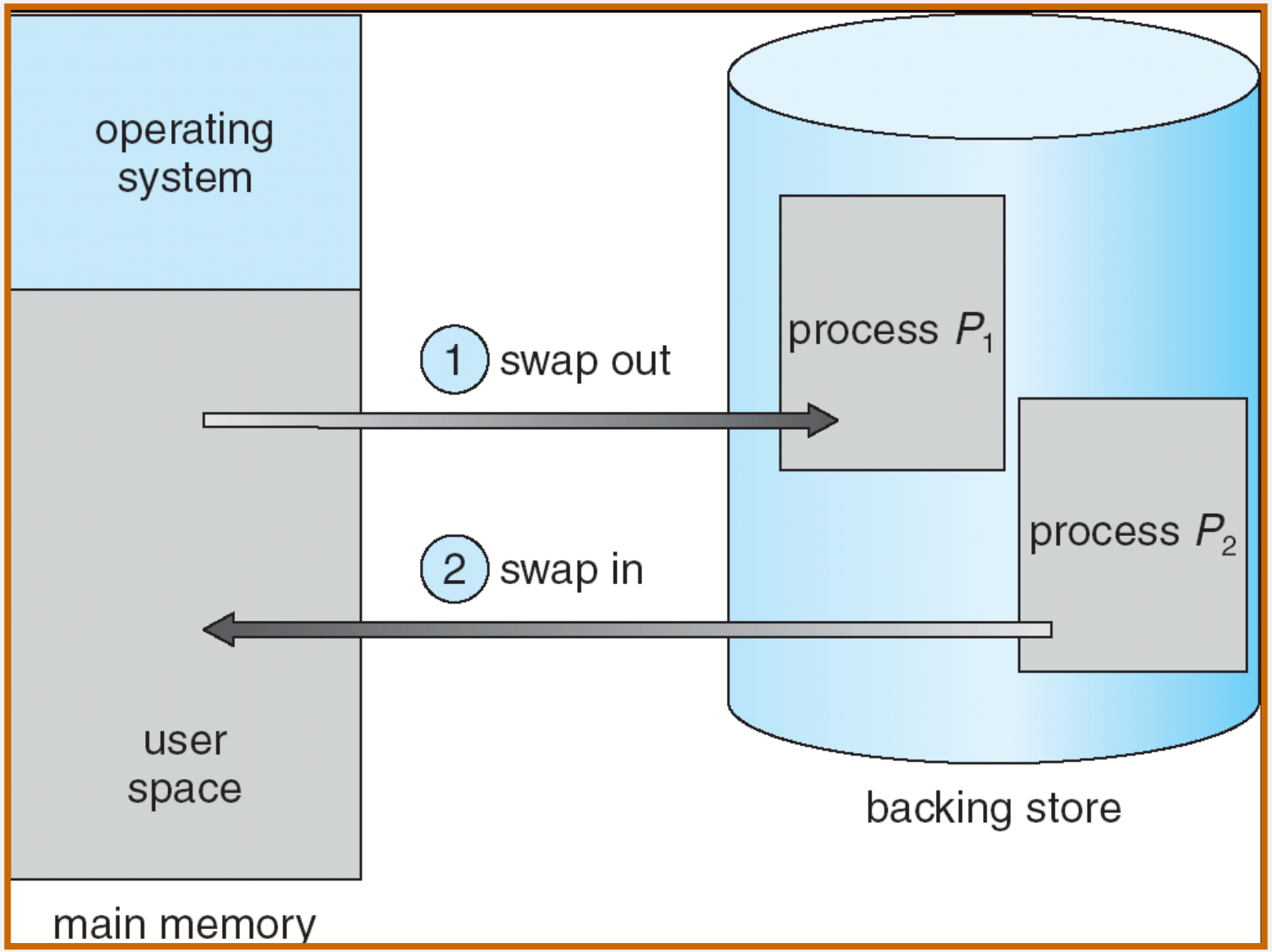
SEGMENTAÇÃO



TROCA DE PROCESSOS (*SWAPPING*)

- Um processo pode ser transferido (*swapped)* temporariamente da memória principal para uma memória secundária (*backing store)*, para depois ser transferido de volta à memória principal, a fim de que a execução do processo continue.
- Processo de baixa prioridade é transferido para a memória secundária para um processo de prioridade mais alta ser carregado e executado.
- Maior parte do tempo de troca de processos é tempo de transferência; tempo total de transferência é diretamente proporcional a quantidade de memória transferida.
- Versões modificadas de *swapping* são encontradas em muitos sistemas (como UNIX, Linux e Microsoft Windows)
- Sistemas mantém uma fila de processos prontos que mantém imagens no disco

VISÃO ESQUEMÁTICA DO *SWAPPING*



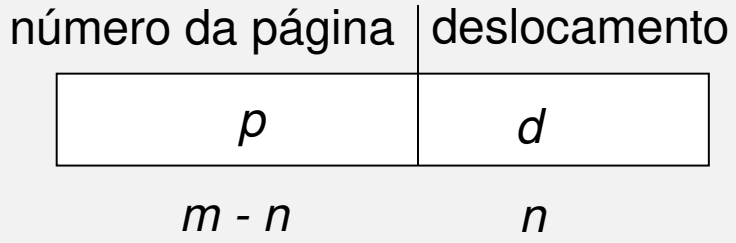
PAGINAÇÃO

- Espaço de endereçamento lógico de um processo pode ser não contíguo; processo é alocado para a memória física sempre que existir espaço disponível
- Divide a memória física em partes de tamanho fixo chamadas de **blocos** (*frames*) (tamanho é potência de 2, entre 512 bytes e 8.192 bytes)
- Divide memória lógica em partes do mesmo tamanho chamadas de **páginas**
- Mantém controle de todos os *blocos* livres
- Para executar um programa com *n* páginas, necessita encontrar *n* blocos livres e carregar o programa
- Alterar uma *tabela de páginas* para traduzir endereços lógicos em físicos

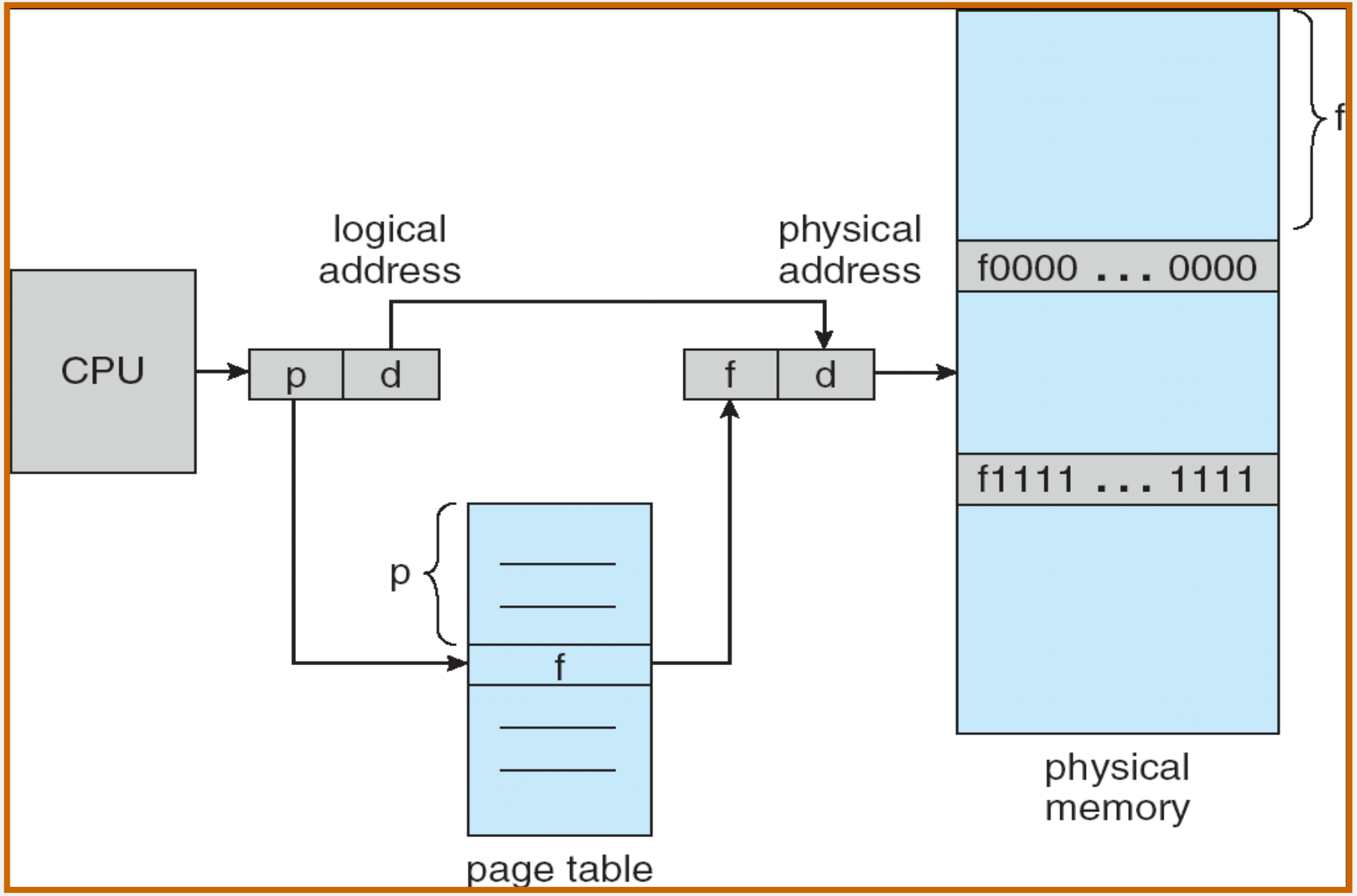
ESQUEMA DE TRADUÇÃO DE ENDEREÇOS

Endereço gerado pela CPU é divido em:

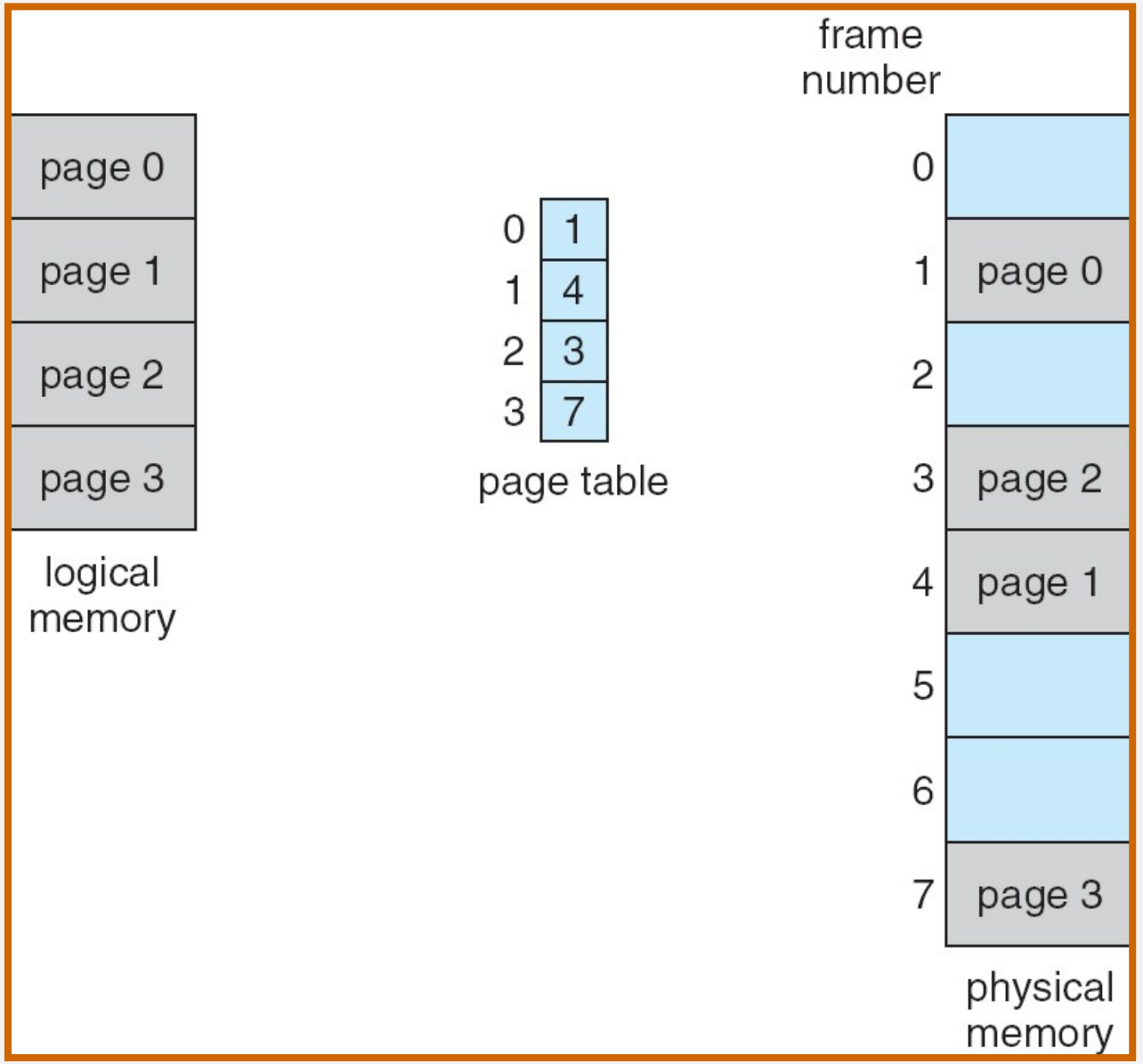
- *Número da Página* (***p***) – usada como um índice em uma tabela de páginas que contém o endereço base de cada página na memória física
- *Deslocamento na Página* (***d***) – combinado com o endereço base para definir o endereço de memória que é enviado a unidade de memória
- Para um determina espaço de endereçamento lógico 2^m e um tamanho de página 2^n



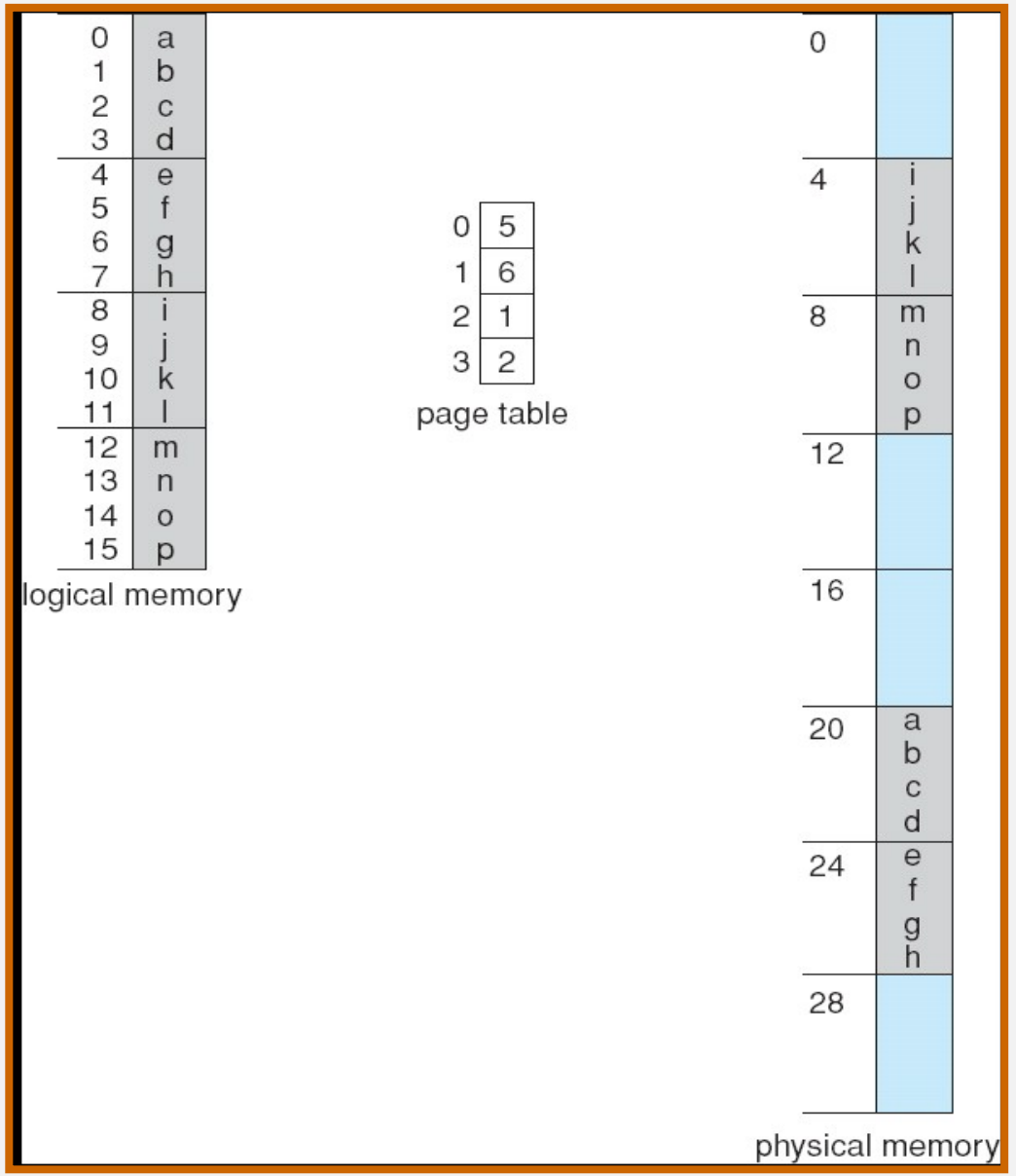
ARQUITETURA PARA TRADUÇÃO DE ENDEREÇOS



EXEMPLO DE PAGINAÇÃO

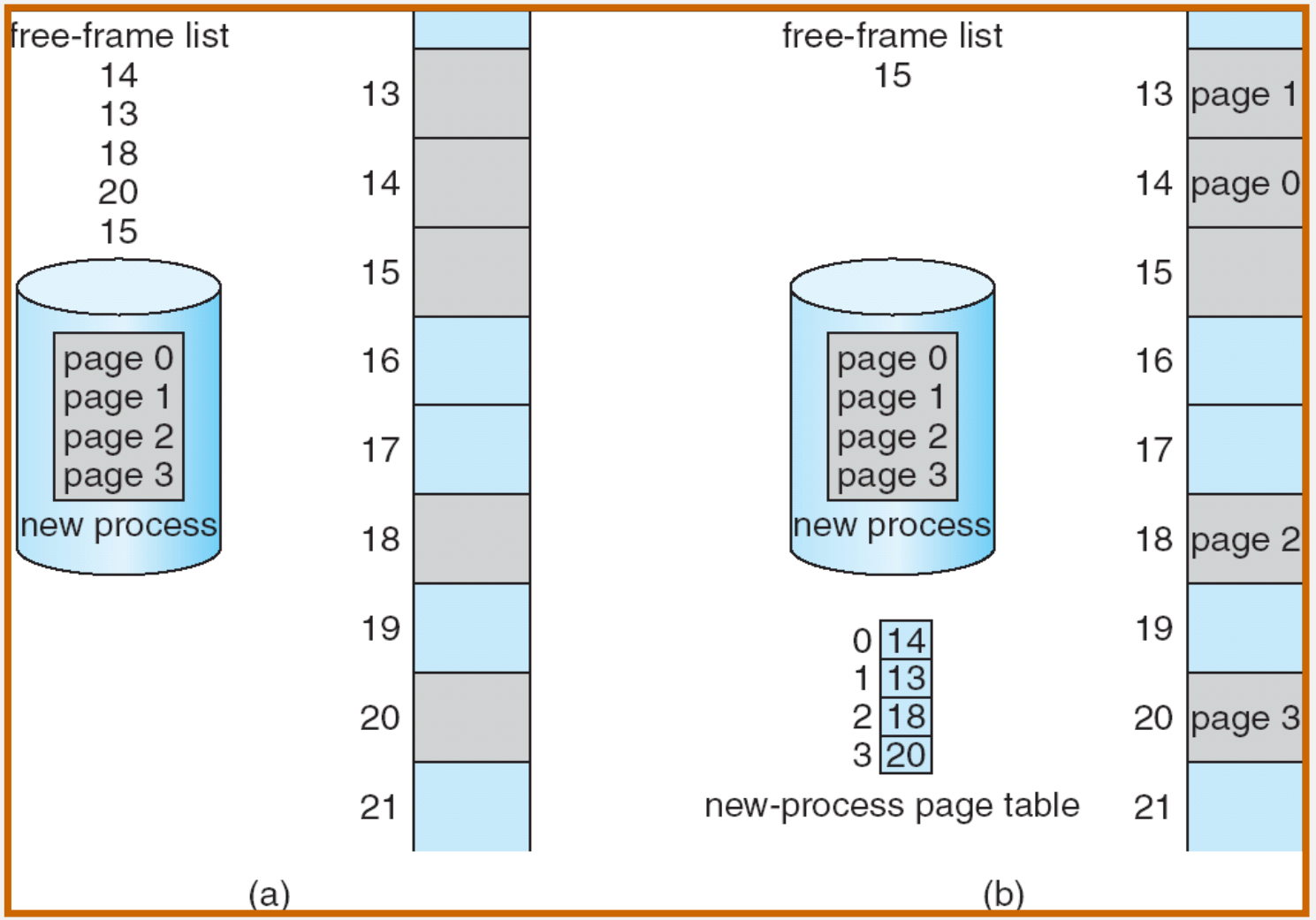


EXEMPLO DE PAGINAÇÃO



Memória de 32 bytes
e páginas de 4 bytes

BLOCOS LIVRES



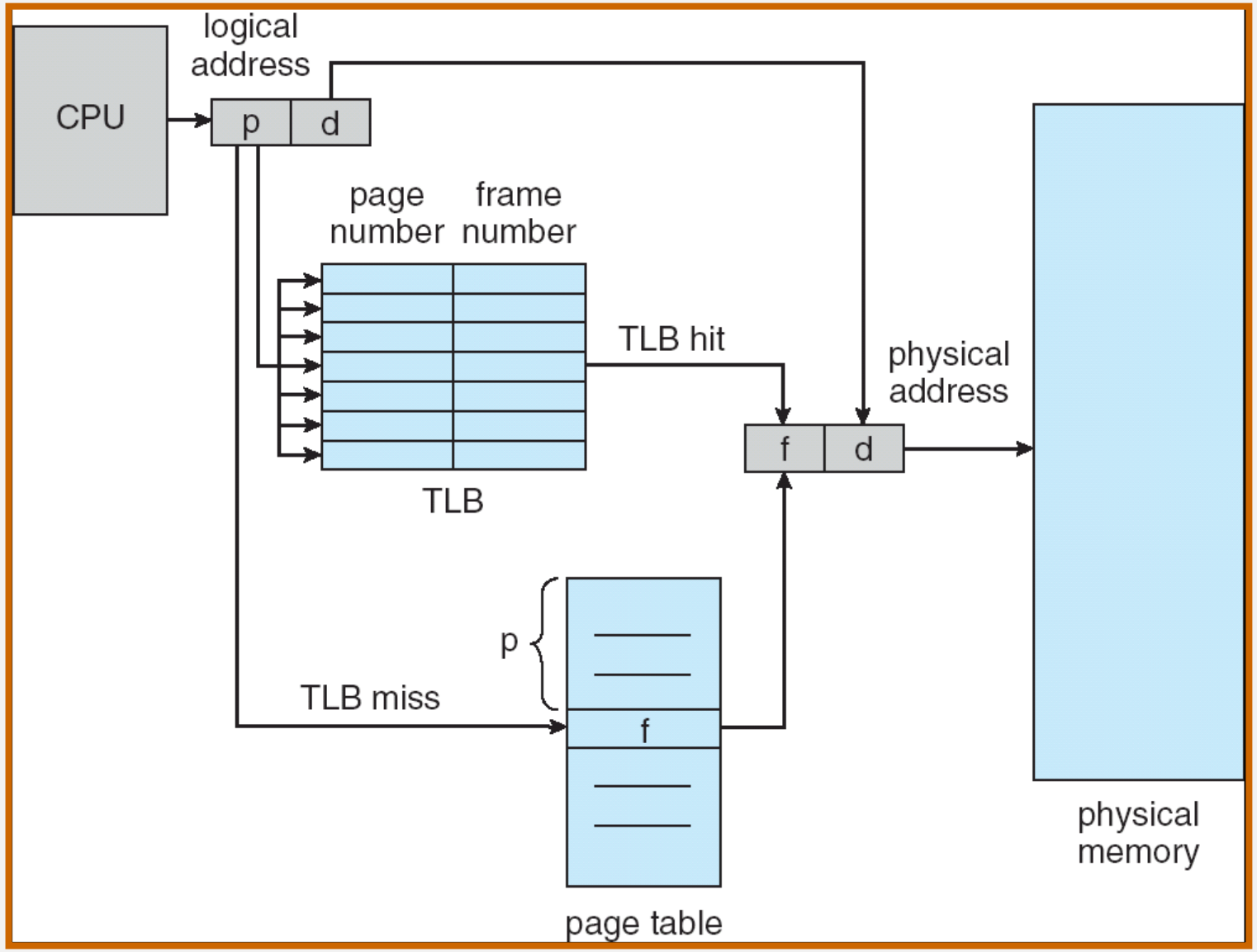
Antes da alocação

Depois da alocação

IMPLEMENTAÇÃO DE TABELA DE PÁGINAS

- Tabela de Páginas é mantida na memória principal.
- ***Registrador base da tabela de páginas*** *(Page-table base register* - **PTBR**) aponta para a tabela de páginas
- ***Registrador tamanho da tabela de páginas*** (*Page-table length register* - **PRLR**) indica quantos endereços ela ocupa
- Neste esquema cada acesso a dado/instrução requer dois acessos a memória. Um para a tabela de páginas e outro para o dado/instrução
- O problema pode ser resolvido com o uso de uma memória *cache* especial, pequena, de acesso rápido, chamada de **memória associativa** ou ***translation look-aside buffers*** (**TLBs**)
- Algumas TLBs armazenam **identificadores de espaços de endereço** (*address-space identifiers* - **ASIDs**) em cada entrada da TLB – identificam cada processo de forma única para prover proteção no espaço de endereçamento daquele processo

HARDWARE DE PAGINAÇÃO COM TLB

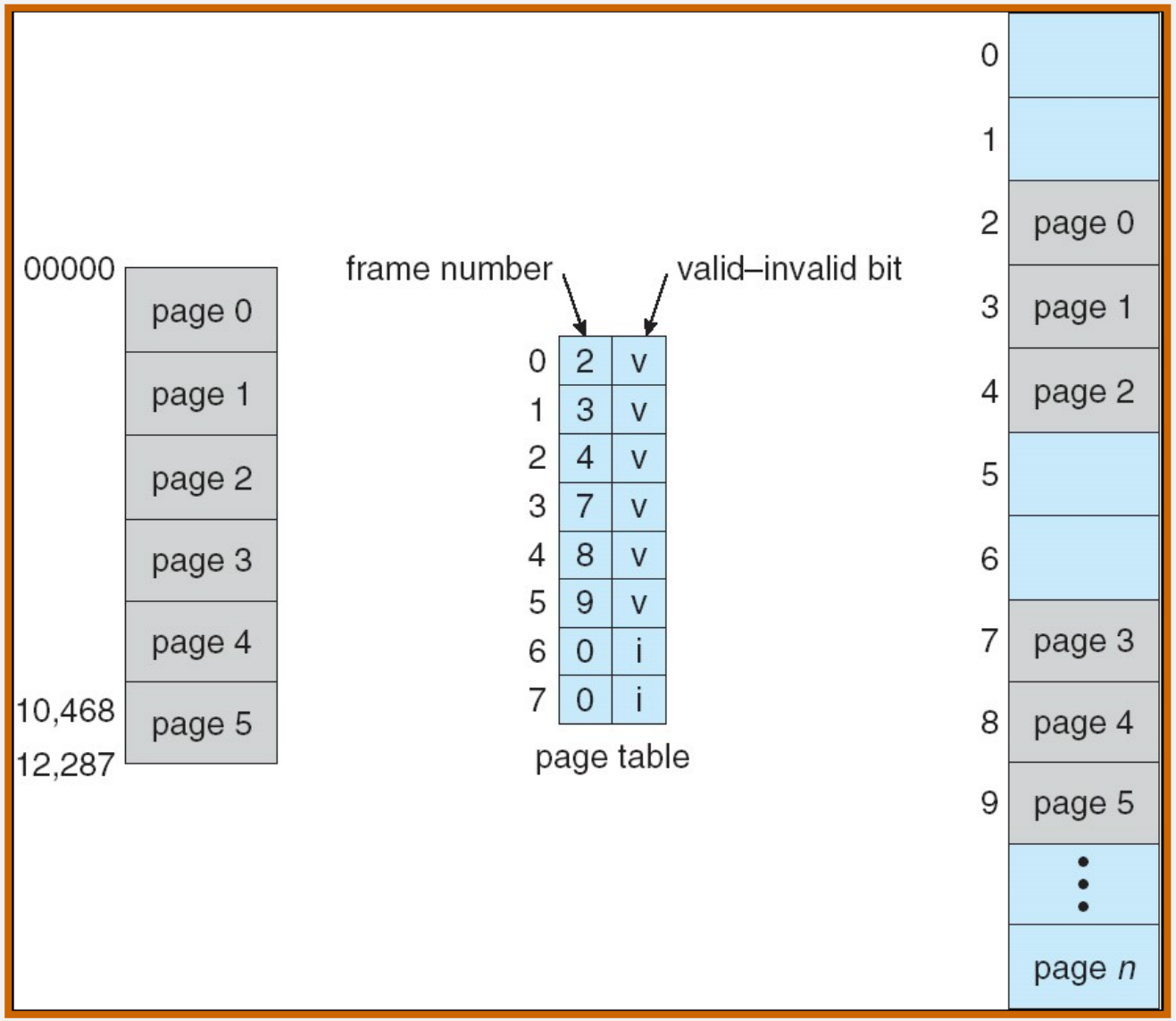


PROTEÇÃO DE MEMÓRIA

- Proteção de Memória implementada através de bits de proteção associados a cada bloco.

- Bit ***válido-inválido*** associado para cada entrada na tabela de páginas:
  - “válido” indica que a página associada está no espaço de endereçamento lógico do processo, e portanto é o acesso é legal.
  - “inválido” indica que a página não está no espaço de endereçamento lógico do processo.

BIT VALIDO (V) OU INVALIDO (I) EM UMA TABELA DE PÁGINAS



MEMÓRIA VIRTUAL

- O conjunto de páginas de um processo presentes na memória é chamado de **conjunto residente**.
  - Inicialmente, algumas páginas são carregadas a partir da página 0.

- Quando é necessário um endereço que não está presente na memória, uma interrupção é gerada:
  - SO coloca processo em estado bloqueado;
  - SO faz pedido de E/S para trazer mais página(s).
  - enquanto isso, SO escolhe outro processo para executar.

MEMÓRIA VIRTUAL

- Quando a página é trazida para a memória, o primeiro processo (o que ocasionou falta de páginas) :
  - passa para a fila dos prontos;
  - controlador envia interrupção (evento realizado).

- Transparente para usuário

- Memória virtual = memória + disco

VANTAGENS

- Mais processos (seus pedaços!) podem estar na memória
  - maior eficiência pois aumenta a chance de um deles estar em estado pronto.

- Processos podem ser maiores que o espaço de endereços físicos (tão grande quanto a área de armazenamento disponível no disco)
  - Responsabilidade do SO e hw trazer partes do processo.

- Reduz o tempo de *swapping*
  - não é toda a imagem que está na memória.
  - somente páginas requisitadas são carregadas.

TRASHING

- Memória contém páginas de diferentes processos.
- Páginas são carregadas, outras são *swapped out:*
  - Possibilidade de enviar para disco uma página de outro processo logo antes desta ser utilizada.
- O processador pode gastar a maior parte do tempo fazendo *swapping* em vez de processando instruções do usuário.
- Como evitar:
  - tentando adivinhar qual página será mais necessária