Sistemas Operacionais

Prof. Jó Ueyama

Apresentação baseada nos slides da Profa. Dra. Kalinka Castelo Branco, do Prof. Dr. Antônio Carlos Sementille e da Profa. Dra. Luciana A. F. Martimiano e nas transparências fornecidas no site de compra do livro "Sistemas Operacionais Modernos"

Gerenciamento de Memória

- Recurso importante;
- Tendência atual do software
  - Lei de *Parkinson*: “Os programas se expandem para preencher a memória disponível para eles” (adaptação);
- Hierarquia de memória:
  - *Cache*;
  - Principal;
  - Disco;

Gerenciamento de Memória

- Idealmente os programadores querem uma memória que seja:
  - Grande
  - Rápida
  - Não Volátil
  - Baixo custo

- Infelizmente a tecnologia atual não comporta tais memórias

- A maioria dos computadores utiliza Hierarquia de Memórias que combina:
  - Uma pequena quantidade de memória cache, volátil, muito rápida e de alto custo
  - Uma grande memória principal (RAM), volátil, com centenas de MB ou poucos GB, de velocidade e custo médios
  - Uma memória secundária, não volátil em disco, com gigabytes (ou terabytes), velocidade e custo baixos

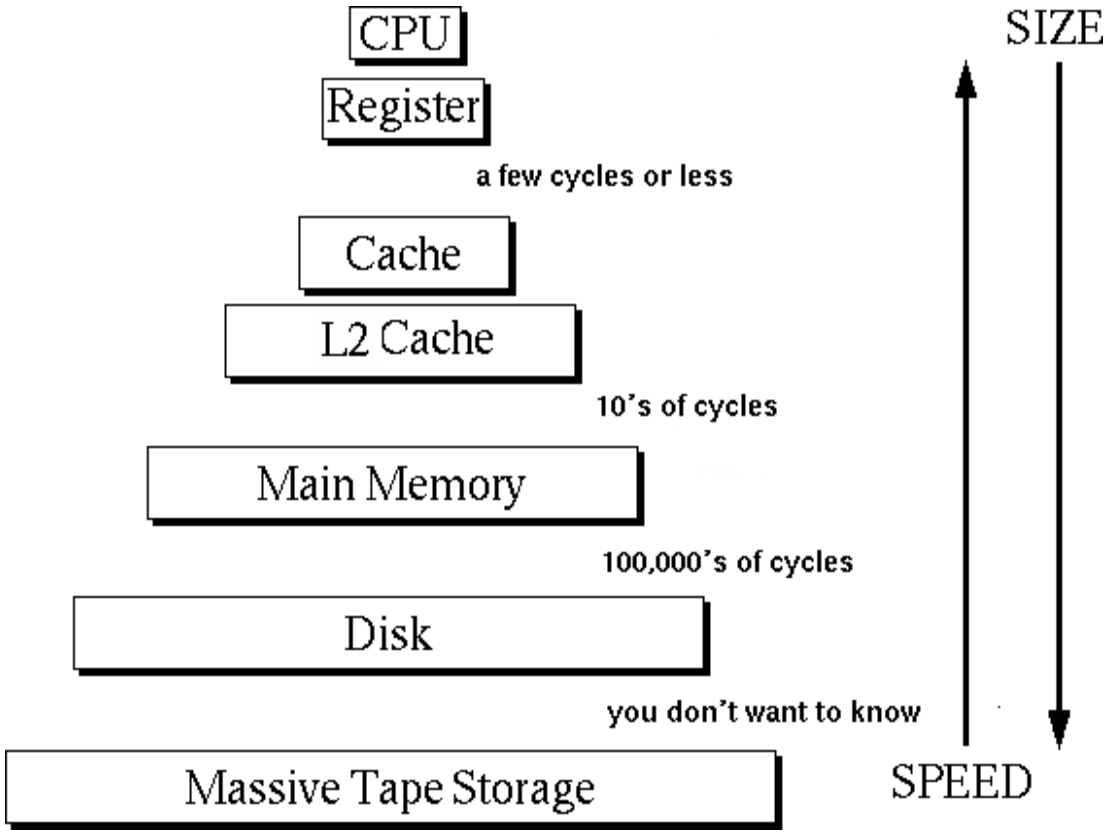
Gerenciamento de Memória
Hierarquia de Memória

- **Cache**
  - Pequena quantidade
    - k bytes
  - Alto custo por byte
  - Muito rápida
  - Volátil

- **Memória Principal**
  - Quantidade intermediária
    - M bytes
  - Custo médio por byte
  - Velocidade média
  - Volátil

- **Disco**
  - Grande quantidade –
    - G bytes
  - Baixo custo por byte
  - Lenta
  - Não volátil

Hierarquia de Memória



Gerenciamento de Memória

- **Cabe ao Gerenciador de Memória gerenciar a hierarquia de memória**

- **Controla as partes da memória que estão em uso e quais não estão, de forma a:**
  - **alocar memória aos processos, quando estes precisarem;**
  - **liberar memória quando um processo termina; e**
  - **tratar do problema do *swapping* (quando a memória é insuficiente).**

Gerenciamento de Memória

- Para cada tipo de memória:
  - Gerenciar espaços livres/ocupados;
  - Alocar processos/dados na memória;
  - Localizar dado;
- Gerenciador de memória:
  - alocar e liberar espaços na memória para os processos em execução;
  - gerenciar chaveamento entre a memória principal e o disco, e memória principal e memória *cache*;

Gerenciamento Básico de Memória

- Sistemas de Gerenciamento de Memória, podem ser divididos em 2 classes:
  - durante a execução levam e trazem processos entre a memória principal e o disco (troca de processos e paginação)
  - mais simples, que não fazem troca de processos e nem paginação

Monoprogramação sem trocas de processos ou Paginação

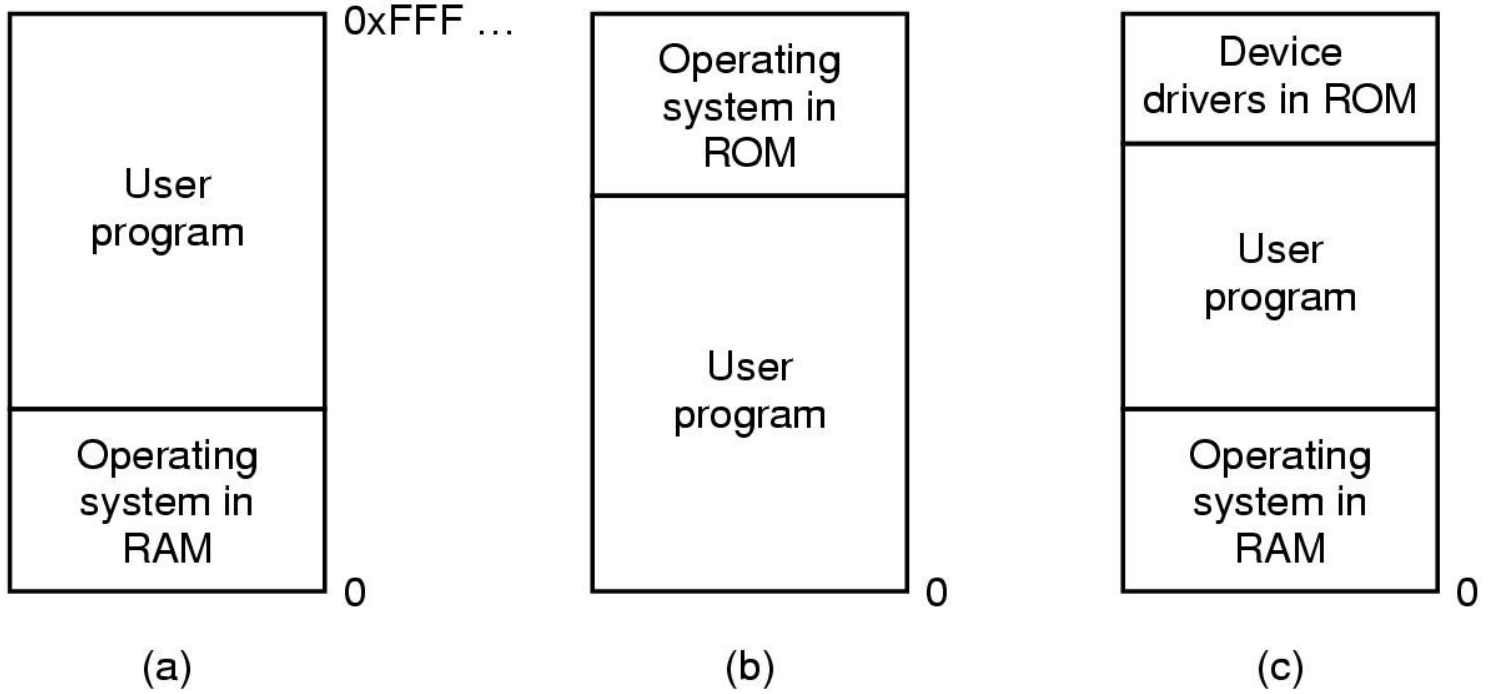
- **Sistemas Mono-usuários: gerência de memória é bem simples, pois toda a memória é alocada à próxima tarefa, incluindo a área do S.O.**

- **Erros de execução podem vir a danificar o S.O.**

- **Neste caso, a destruição do S.O. é um pequeno inconveniente, resolvido pelo recarregamento do mesmo.**

Gerenciamento Básico de Memória

Monoprogramação sem trocas de processos ou Paginação



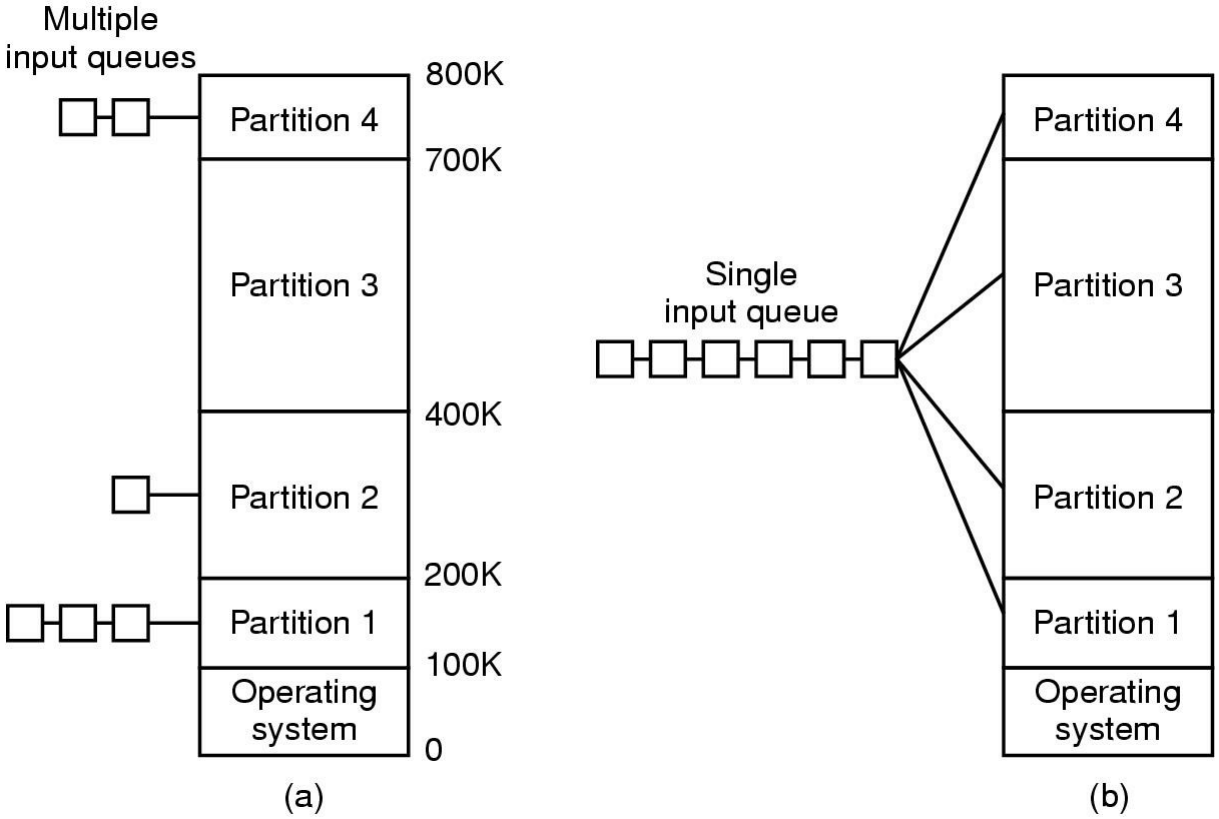
Três esquemas simples de organização de memória

- Um sistema operacional com um processo de usuário

Multiprogramação com partições Fixas

- **Sistemas Monoprogramados:** raramente usados atualmente.
- **Sistemas modernos:** permitem multiprogramação
- A maneira mais comum de realizar a multiprogramação é dividir simplesmente a memória em **n partições** (provavelmente de **tamanhos diferentes**).
- Esta divisão pode ser feita de maneira manual, quando o sistema é inicializado
- Ao chegar, um *job*, pode ser colocado em uma fila de entrada associada à menor partição, grande o suficiente para armazená-lo

Multiprogramação com partições Fixas



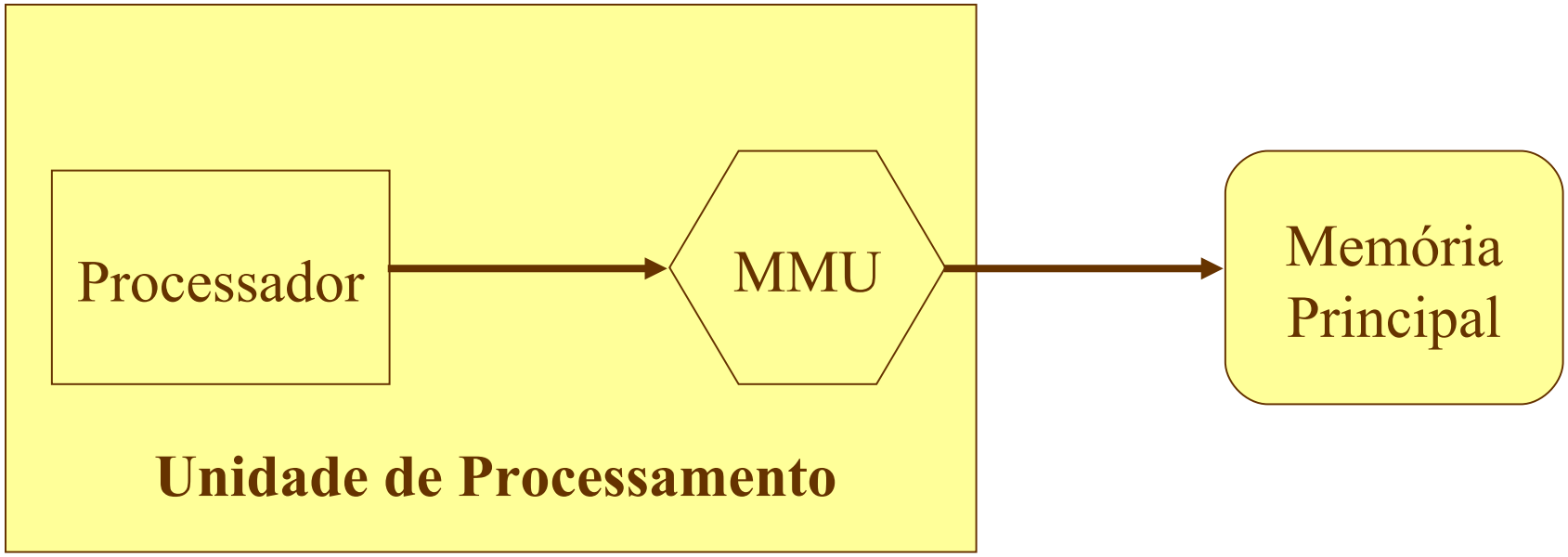
- Partições de memória fixa
  - fila separada para cada partição
  - uma única fila de entrada

Gerenciamento de Memória

- Multiprogramação → Vários processos na memória:
  - Como proteger os processos uns dos outros e o *kernel* de todos os processos?
  - Como tratar a realocação?
- Todas as soluções envolvem equipar a CPU com um hardware especial → MMU (*memory management unit*);

Gerenciamento de Memória

- MMU – *Memory Management Unit*



Gerenciamento de Memória

- **MMU** (do inglês **Memory Management Unit**) é um dispositivo de hardware que transforma endereços virtuais em endereços físicos.
- Na MMU, o valor no registro de re-locação é adicionado a todo o endereço lógico gerado por um processo do utilizador na altura de ser enviado para a memória.
- O programa do utilizador manipula endereços lógicos; ele nunca vê endereços físicos reais.

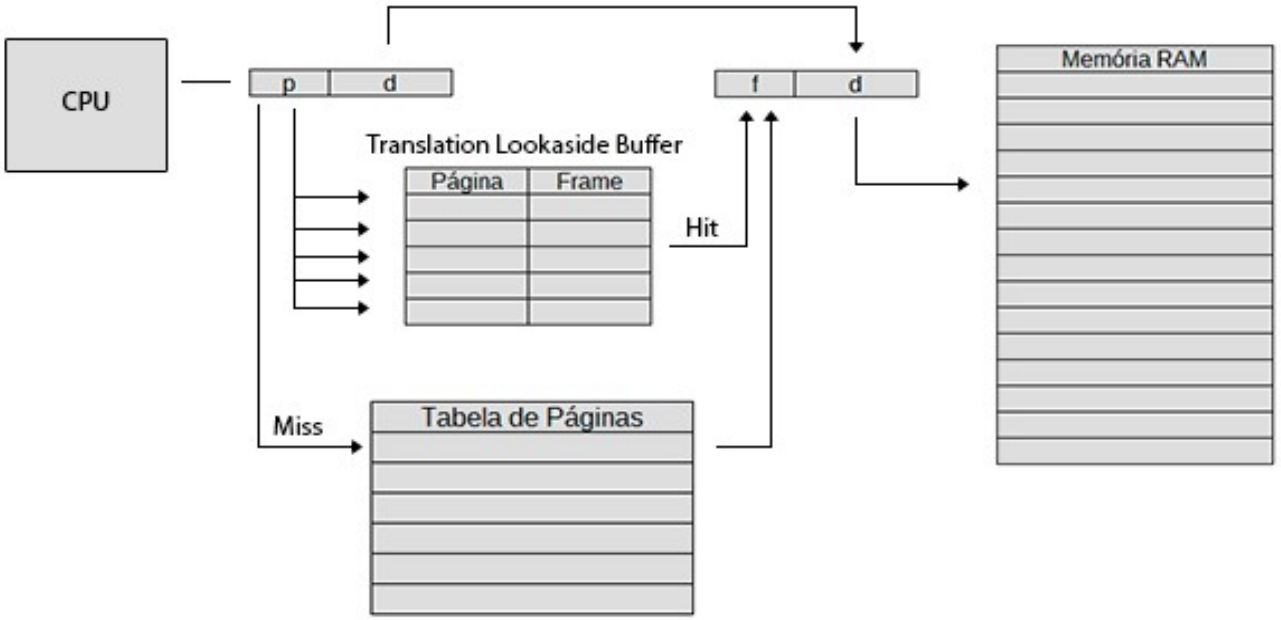
Gerenciamento de Memória

- Normalmente o sistema atual de MMU divide o espaço de endereçamento virtual (endereços utilizados pelo processador) em páginas, cujo o tamanho é de 2^n, tipicamente poucos ***kilobytes***.

- A MMU normalmente traduz número de páginas virtuais para número de páginas físicas utilizando uma *cache* associada chamada Translation Lookaside Buffer (TLB)

- Quando o TLB falha uma tradução, um mecanismos mais lento envolvendo um *hardware* específico de dados estruturados ou um *software* auxiliar é usado.

TLB – Translation Looaside Buffer



Relocação e Proteção

- Relocar é deslocar um código de um local para outro
- Pode não ter certeza de onde o programa será carregado na memória
  - As localizações de endereços de localização das variáveis e do código das rotinas não podem ser absolutos
  - Tipos: estático (muda o código antes de executar) e dinâmico (relocação feita na execução através de regs)

- Uso de valores de base e limite
  - Os endereços das localizações são somados a um valor de base para mapear um endereço físico
  - Valores de localizações maiores que um valor limite são considerados erro

Relocação e proteção

- Realocação:
  - Quando um programa é *linkado* (programa principal + rotinas do usuário + rotinas da biblioteca → executável) o *linker* deve saber em que endereço o programa irá iniciar na memória;
  - Nesse caso, para que o *linker* não escreva em um local indevido, como por exemplo na área do SO (100 primeiros endereços), é preciso de realocação:
    - #100 + ! → que depende da partição!!!

Relocação e proteção

- Proteção:
  - Com várias partições e programas ocupando diferentes espaços da memória é possível acontecer um acesso indevido;

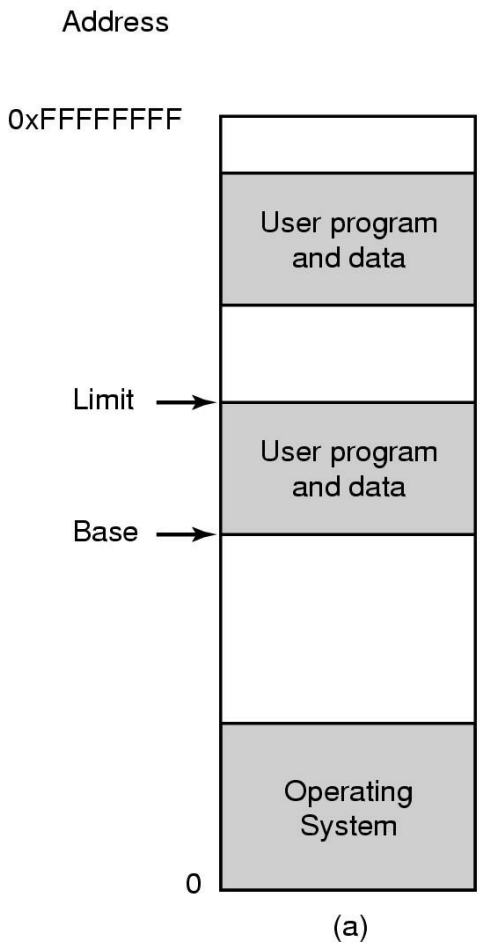
- Solução para ambos os problemas:
  - 2 registradores → base e limite
    - Quando um processo é escalonado o registrador-base é carregado com o endereço de início da partição e o registrador-limite com o tamanho da partição;
    - O registrador-base torna impossível a um processo uma remissão a qualquer parte de memória abaixo de si mesmo.

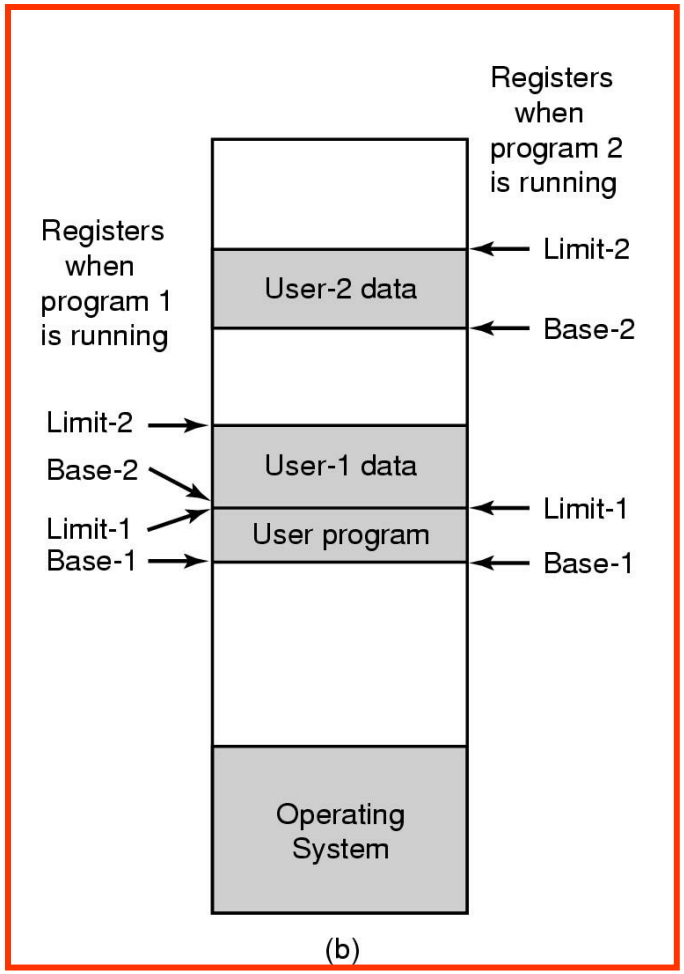
Relocação e proteção

- 2 registradores → base e limite
  - Automaticamente, a MMU adiciona o conteúdo do registrador-base a cada endereço de memória gerado;
  - Endereços são comparados com o registrador-limite para prevenir acessos indevidos;

Gerenciamento de Memória

Registradores base e limite





b) MMU mais sofisticada → dois pares de registradores: segmento de dados usa um par separado;

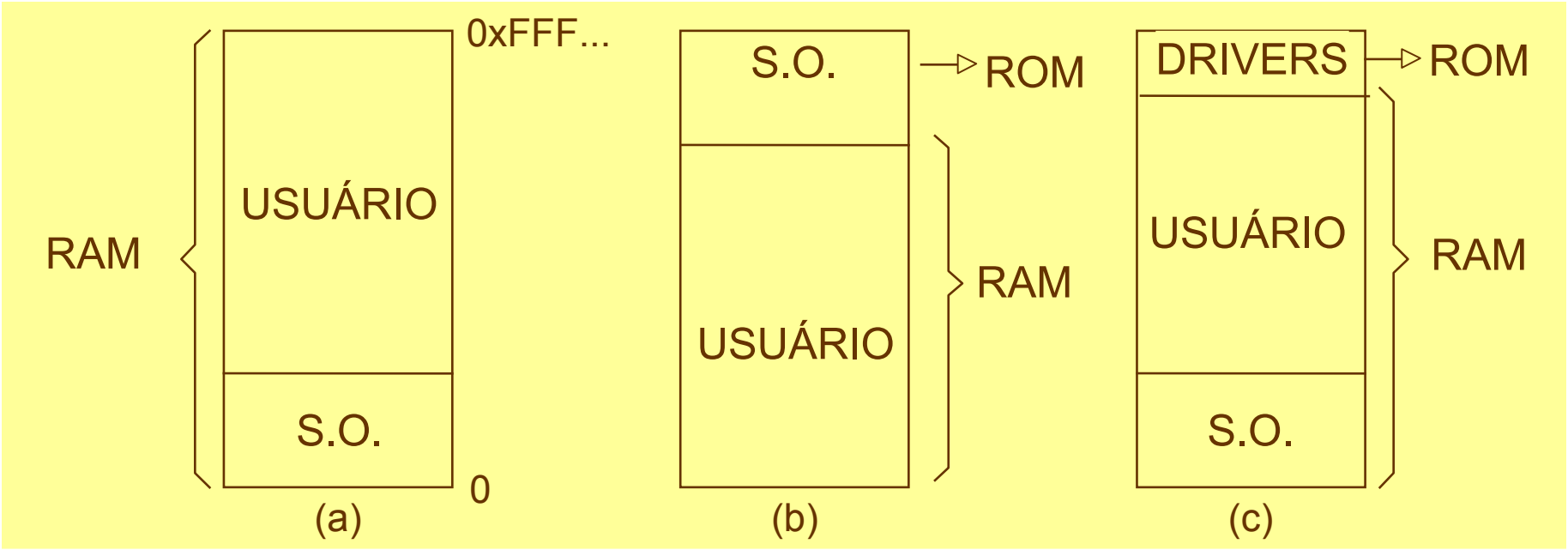
- MMU modernas têm mais pares de registradores.

Gerenciamento de Memória

- Tipos básicos de gerenciamento:
  - Com paginação (chaveamento): Processos são movidos entre a memória principal e o disco; artifício usado para resolver o problema da falta de memória;
    - Se existe MP suficiente não há necessidade de se ter paginação;
  - Sem paginação: não há chaveamento;

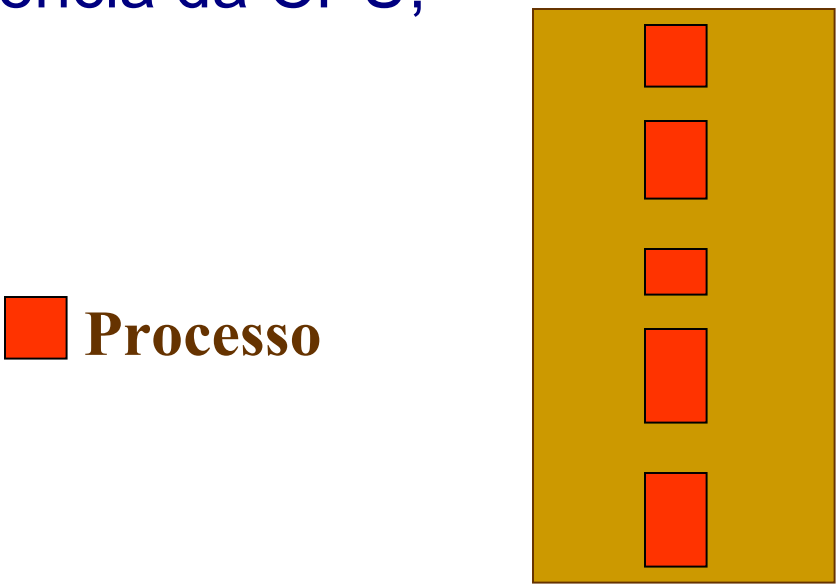
Gerenciamento de Memória

- Monoprogramação:
  - Sem paginação: gerenciamento mais simples;
- Apenas um processo na memória;



Gerenciamento de Memória

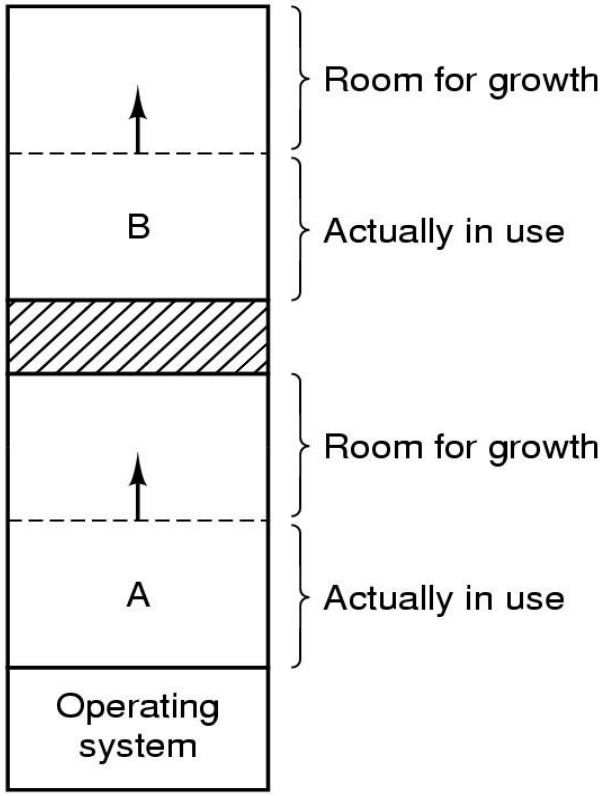
- Modelo de Multiprogramação:
  - Múltiplos processos sendo executados;
  - Eficiência da CPU;



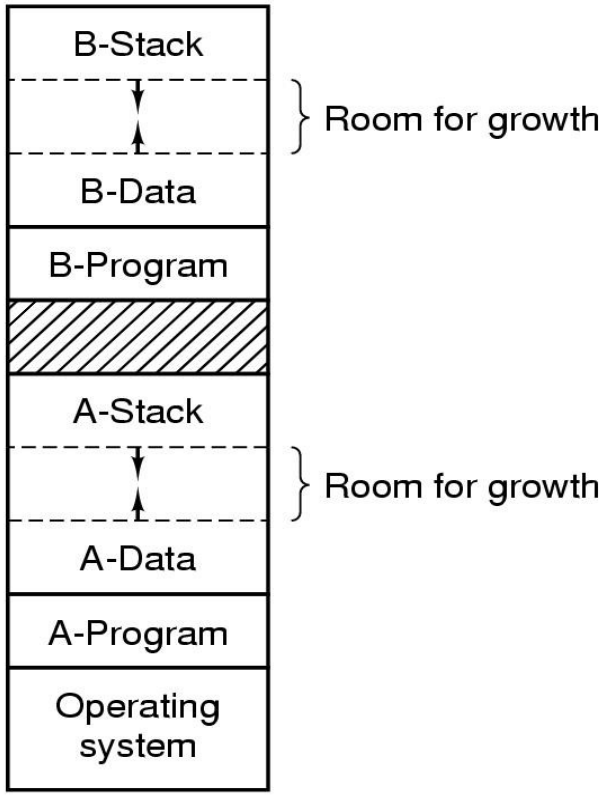
Gerenciamento de Memória
Alocando memória

a) segmento de dados;

b) segmento de dados e de pilha;



(a)



(b)

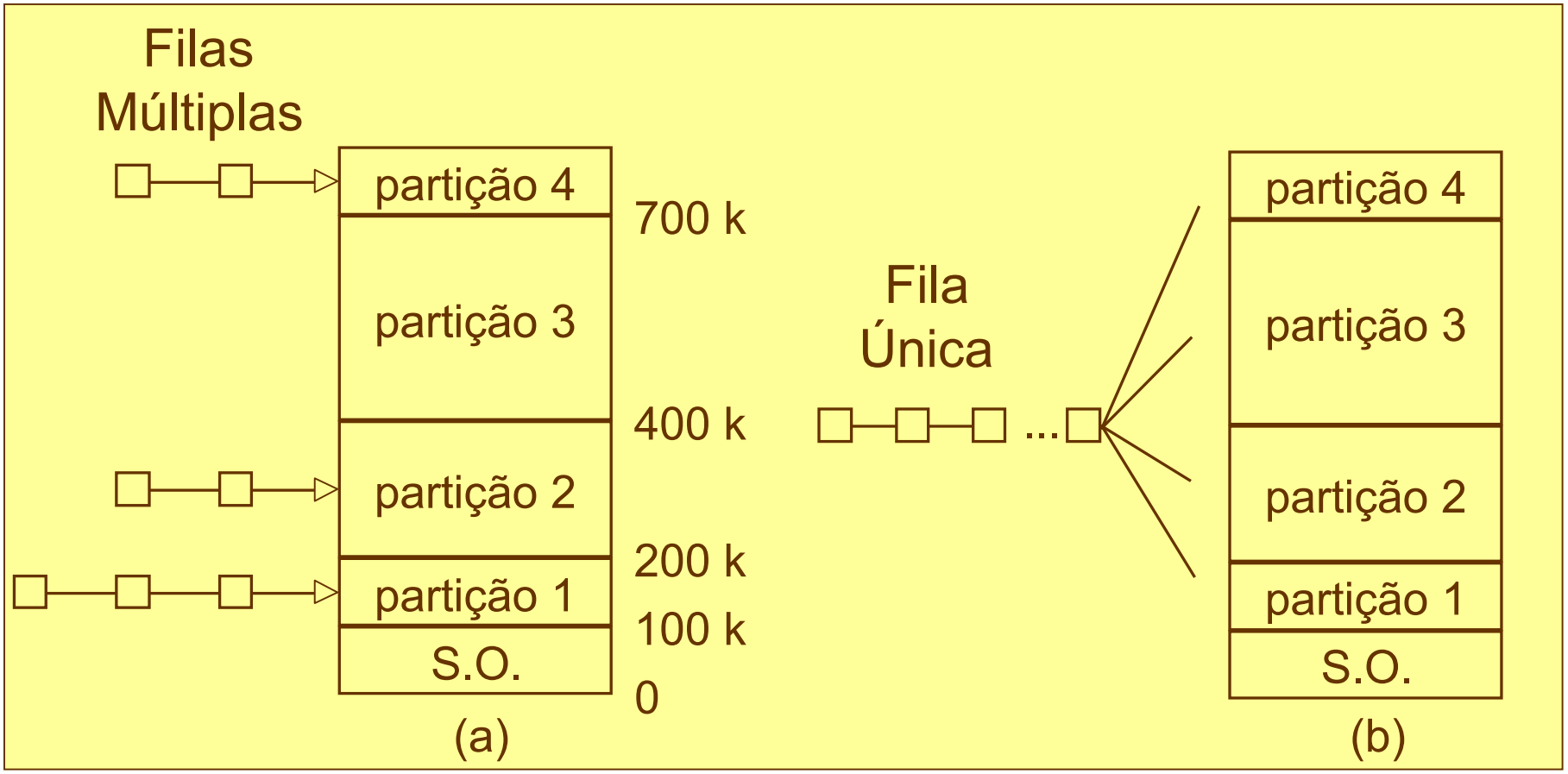
Gerenciamento de Memória
Partições/Alocação

- Particionamento da memória pode ser realizado de duas maneiras:
  - Partições fixas (ou alocação estática);
  - Partições variáveis (ou alocação dinâmica);
- **Partições Fixas**:
  - Tamanho e número de partições são fixos (estáticos);
  - Não é atrativo, porque partições fixas tendem a desperdiçar memória (Qualquer espaço não utilizado é literalmente perdido)
  - Mais simples;

Gerenciamento de Memória
Partições Fixas

- Partições Fixas:
  - Filas múltiplas:
    - Problema: filas não balanceadas;
  - Fila única:
    - Facilita gerenciamento;
    - Implementação com Lista:
      - Melhor utilização da memória, pois procura o melhor processo para a partição considerada;
      - Problema: processos menores são prejudicados;

Gerenciamento de Memória
Partições Fixas



Gerenciamento de Memória
Partições Fixas

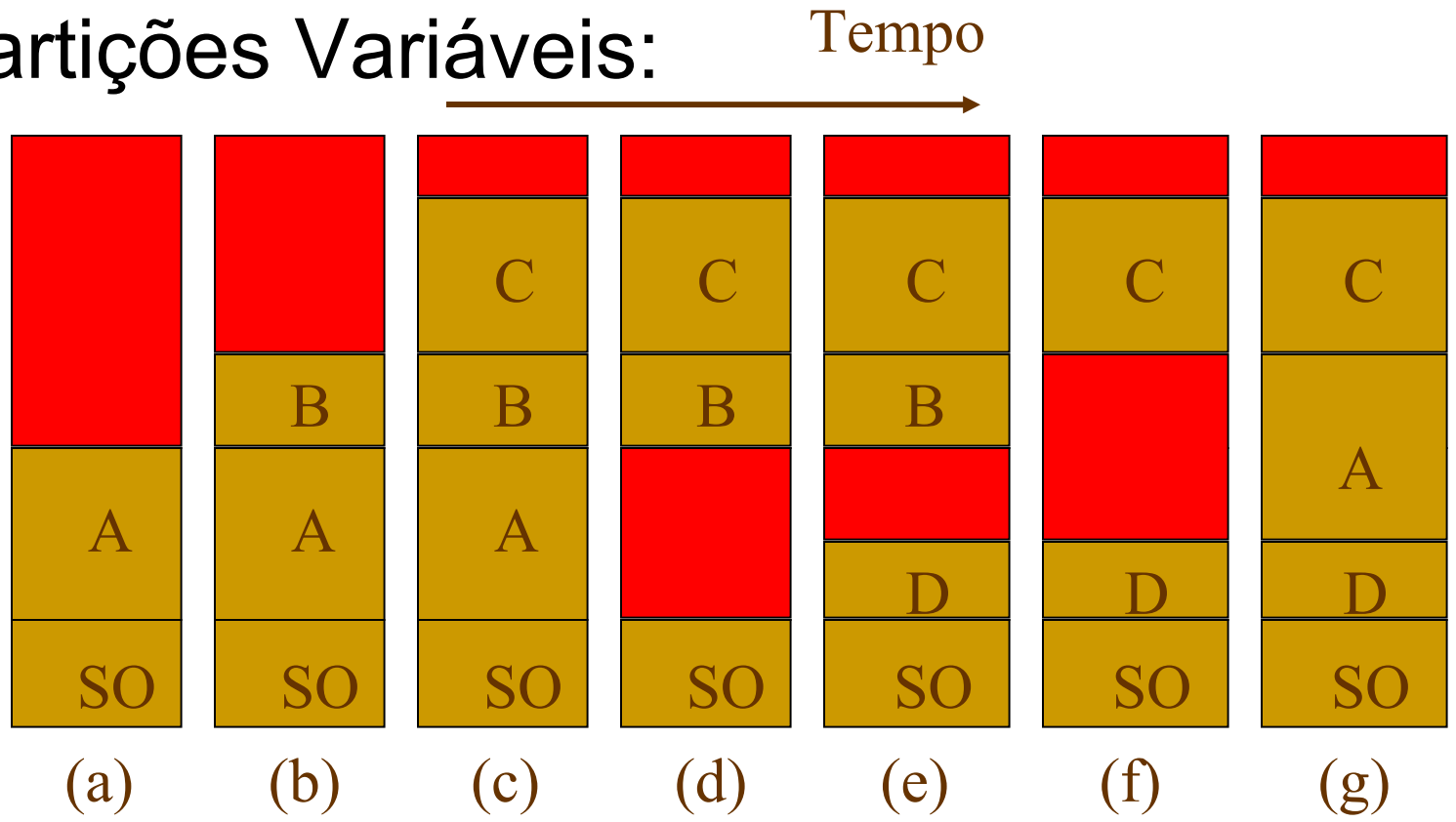
- Partições Fixas: problemas com fragmentação:
  - **Interna**: desperdício dentro da área alocada para um processo;
    - Ex.: processo de tamanho 40K ocupando uma partição de 50k;
  - **Externa**: desperdício fora da área alocada para um processo;
    - Duas partições livres: PL1 com 25k e PL2 com 100k, e um processo de tamanho 110K para ser executado;
    - Livre: 125K, mas o processo não pode ser executado;

Gerenciamento de Memória
Partições Variáveis

- **Partições Variáveis**:
  - Tamanho e número de partições variam;
  - Otimiza a utilização da memória, mas complica a alocação e liberação da memória;
  - Partições são alocadas dinamicamente;
  - SO mantém na memória uma lista com os espaços livres;
  - Menor fragmentação interna e grande fragmentação externa. Por que?
    - Solução: Compactação;

Gerenciamento de Memória
Partições Variáveis

- Partições Variáveis:



Troca de Processos

Com um sistema em lote, é simples e eficiente organizar a memória em partições fixas.

Em sistemas de tempo compartilhado:

- Pode não existir memória suficiente para conter todos os processos ativos
- Os processos excedentes são mantidos em disco e trazidos dinamicamente para a memória a fim se serem executados

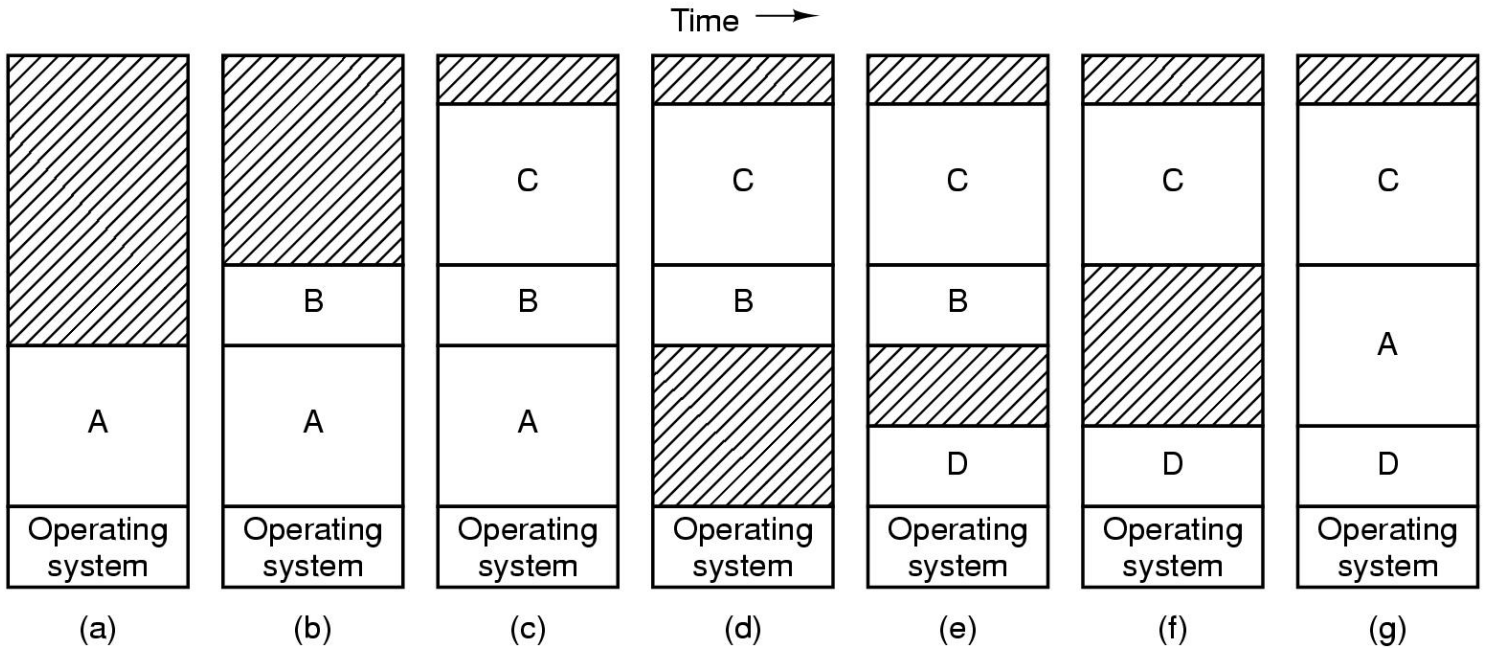
Troca de Processos

Existem 2 maneiras gerais que podem ser usados:

- A troca de processos (*swapping*): forma mais simples, consiste em trazer totalmente cada processo para a memória, executá-lo durante um tempo e, então, devolvê-lo ao disco

- Memória Virtual: permite que programas possam ser executados mesmo que estejam parcialmente carregados na memória principal.

Troca de Processos



A Alocação de memória muda a medida que

- Os processos chegam à memória
- Os processos deixam a memória

As regiões sombreadas (na Figura) representam a memória não usada

Gerenciamento de Memória

- Minimizar espaço de memória inutilizados:
  - Compactação: necessária para recuperar os espaços perdidos por fragmentação; no entanto, muito custosa para a CPU;
- Técnicas para alocação dinâmica de memória:
  - *Bitmaps*;
  - Listas Encadeadas;

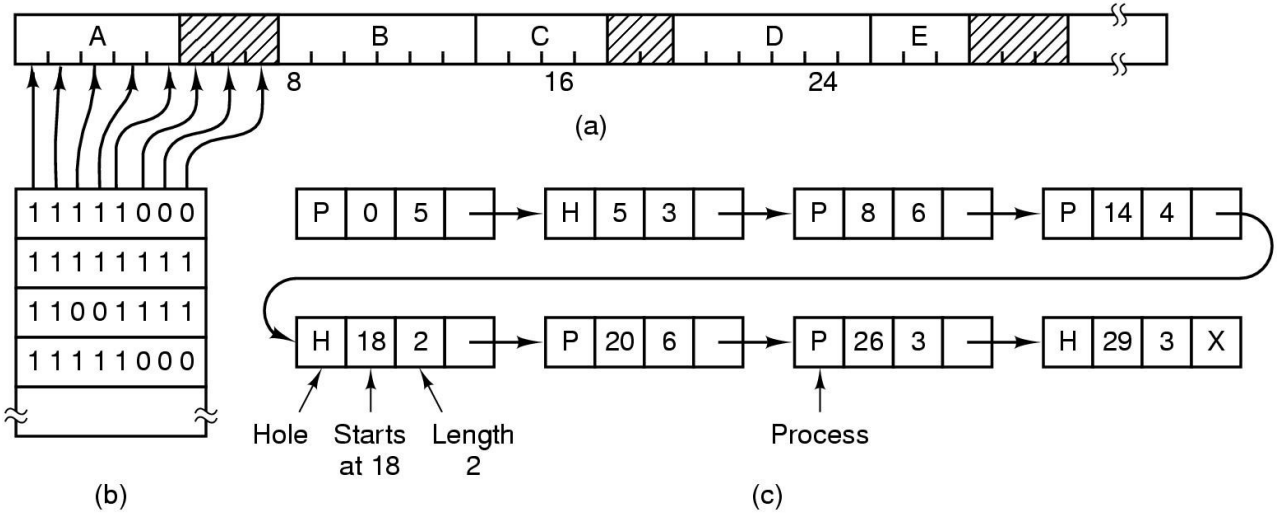
Gerenciamento de Memória

- Técnica com *Bitmaps*:
  - Memória é dividida em unidades de alocação em kbytes;
  - Cada unidade corresponde a um *bit* no *bitmap*:

    0 → livre

    1 → ocupado
  - Tamanho do *bitmap* depende do tamanho da unidade e do tamanho da memória;
  - Ex.:
    - unidades de alocação pequenas → *bitmap* grande;
    - unidades de alocação grandes → perda de espaço;

Gerenciamento de memória com Mapas de Bits



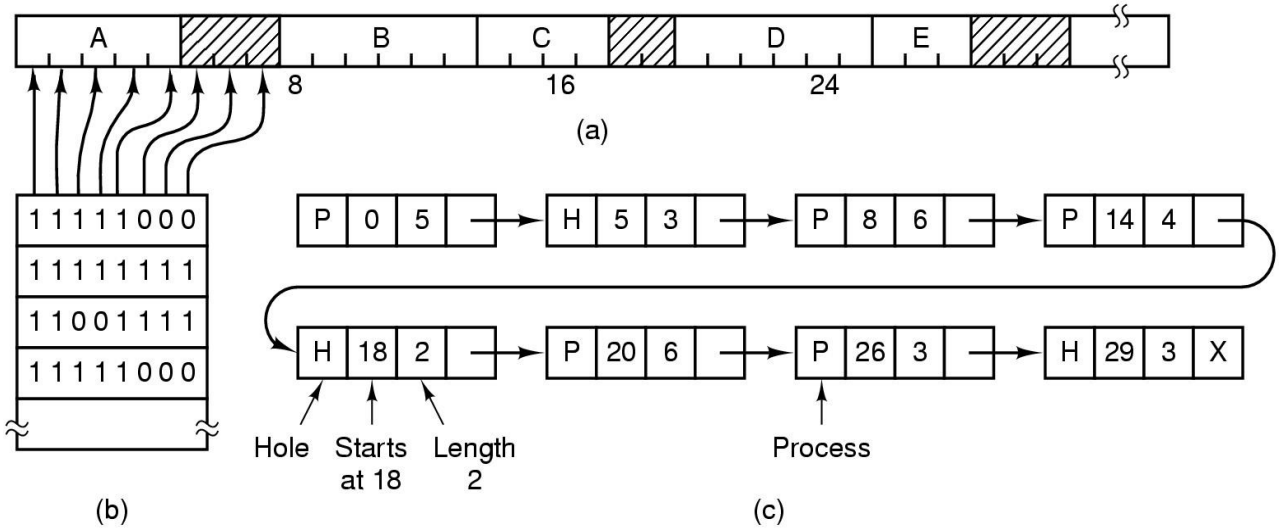
(a) Uma parte da memória com 5 processos e 3 buracos
  - As regiões em branco (1 no bitmap) marcam as unidades já alocadas
  - As regiões sombreadas (0 no bitmap) marcam unidades desocupadas
(b) O Bitmap correspondente
(c) A mesma informação como uma lista encadeada

Gerenciamento de Memória com Listas encadeadas

Outra maneira de gerenciar memória é manter uma lista encadeada de segmentos de memória alocados e segmentos disponíveis

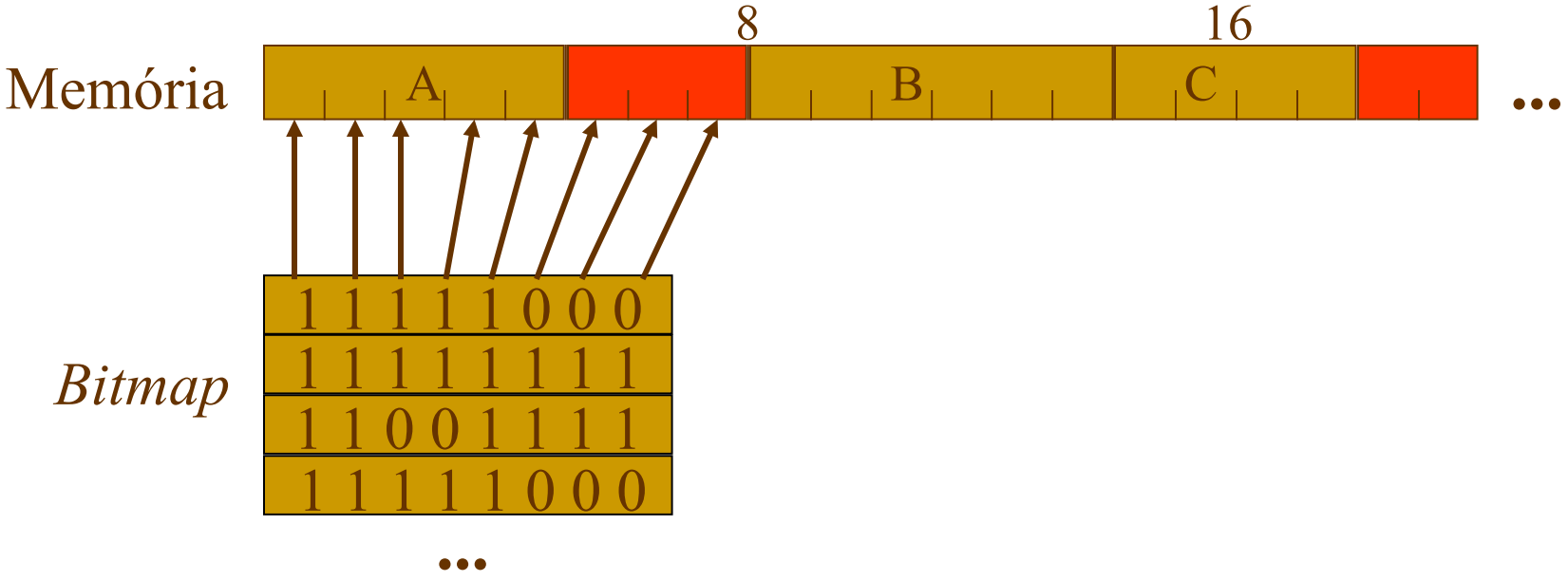
Cada elemento desta lista especifica:

- um segmento disponível (H), ou alocado a um processo (P),
- o endereço onde se inicia este segmento
- e um ponteiro para o próximo elemento



Gerenciamento de Memória

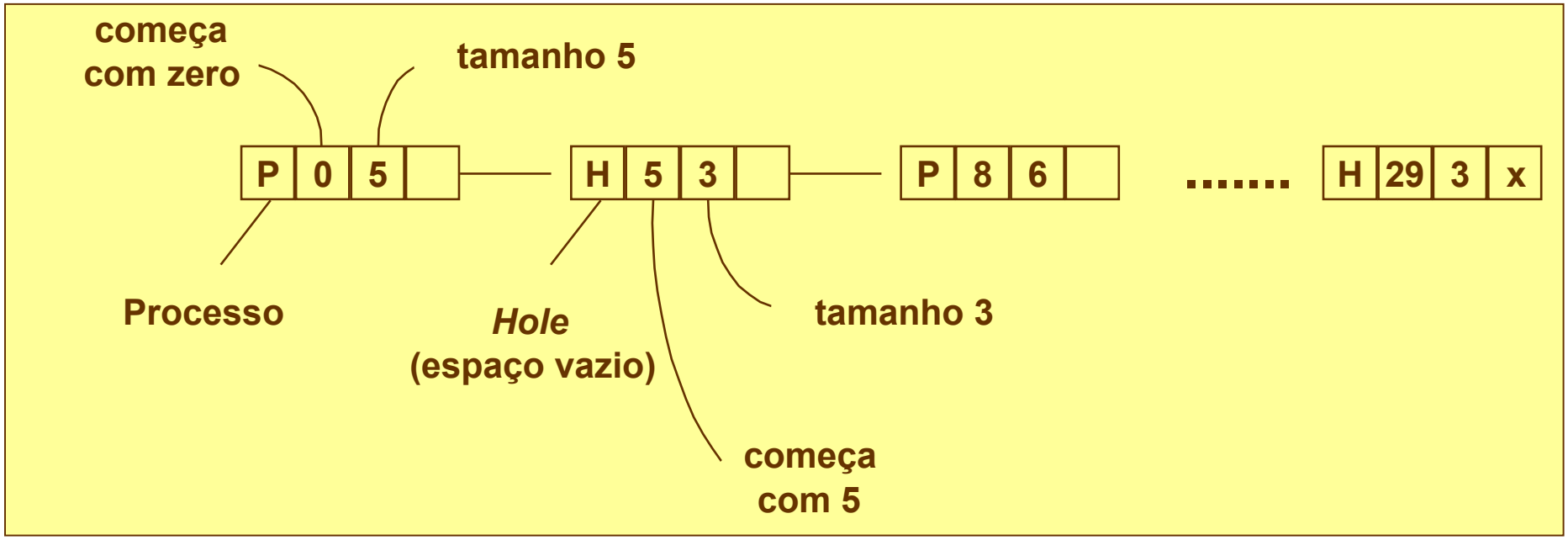
- Técnica com *Bitmaps*:



Memória ocupada
Memória livre

Gerenciamento de Memória

- Técnica com Listas Encadeadas:
  - Uma lista para os espaços vazios e outra para os espaços cheios, ou uma lista para ambos!



Gerenciamento de Memória

Alocação de segmentos livres

- **Existem três métodos que podem ser usados para selecionar uma região para um processo. Os algoritmos de alocação são:**
  - **Melhor escolha (Best fit): colocar o processo no bloco com o mínimo resto de memória;**
  - **Pior escolha (worst fit): usar o bloco com o maior resto de memória;**
  - **Primeira escolha (First fit): ir sequencialmente através dos blocos, e tomar o primeiro grande o suficiente.**

Gerenciamento de Memória

- Processos e espaços livres informados em uma lista encadeada
- Algoritmos de Alocação → quando um novo processo é criado:
  - *FIRST FIT*
    - 1º segmento é usado;
    - Rápido, mas pode desperdiçar memória por fragmentação;
  - *NEXT FIT*
    - 1º segmento é usado;
    - Mas na próxima alocação inicia busca do ponto que parou anteriormente;
    - SIMULAÇÕES: desempenho ligeiramente inferior;

Gerenciamento de Memória

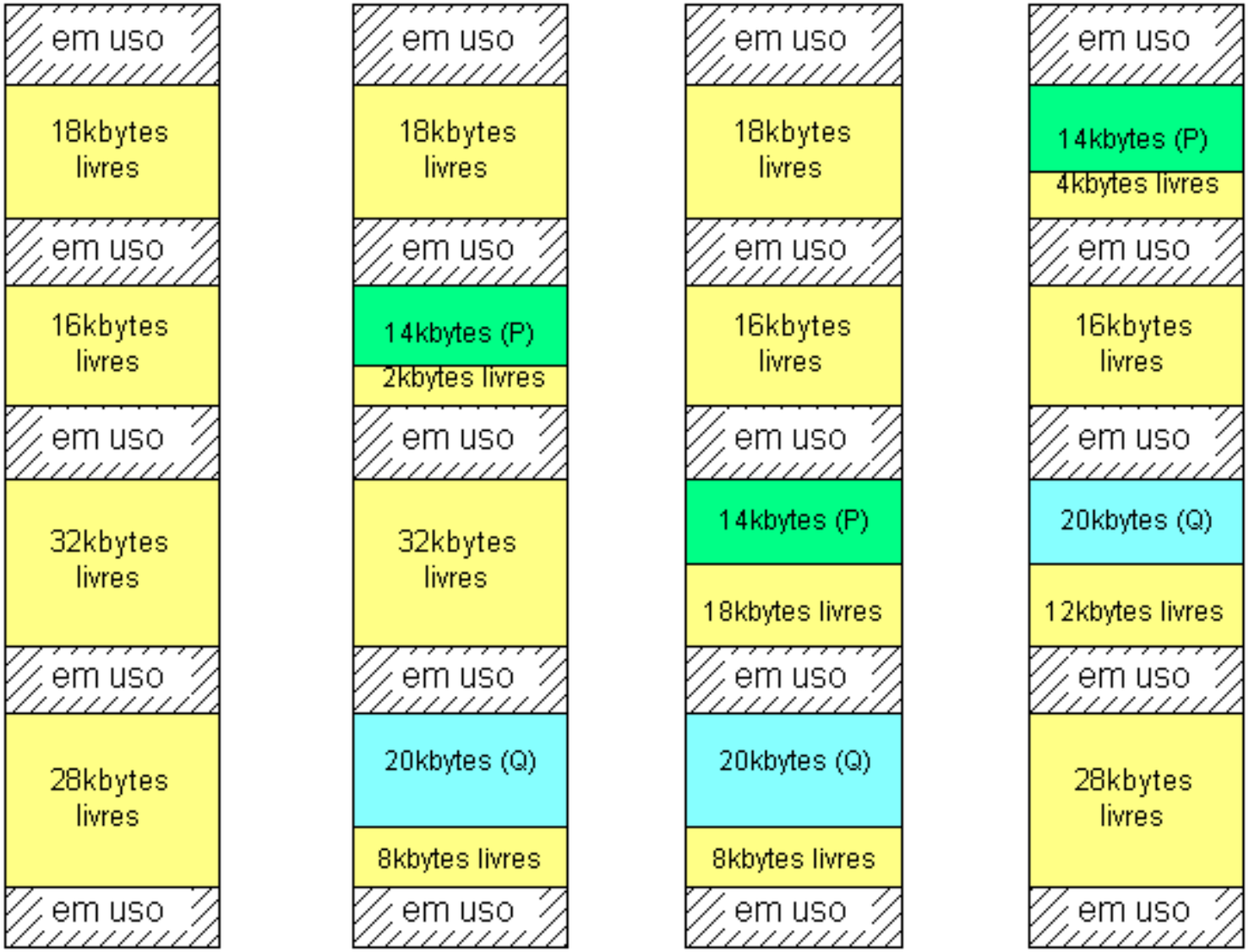
- *BEST FIT*
  - Procura na lista toda e aloca o espaço que mais convém;
  - Menor fragmentação;
  - Mais lento;
- *WORST FIT*
  - Aloca o maior espaço disponível;
- *QUICK FIT*
  - Mantém listas separadas para os espaços mais requisitados;
  - Pode criar uma lista separada para espaços livres de 4KB, 8KB, 12KB
  - Espaços livres de 21KB podem estar na lista de 20KB ou em uma lista de espaços livres de tamanhos especiais

Gerenciamento de Memória

- Cada algoritmo pode manter listas separadas para processos e para espaços livres:
  - Vantagem:
    - Aumenta desempenho;
  - Desvantagens:
    - Aumenta complexidade quando espaço de memória é liberado – gerenciamento das listas;
    - Fragmentação;

Gerenciamento de Memória

Alocação de segmentos livres



Áreas livres iniciais **Melhor Escolha** **Pior Escolha** **Primeira Escolha**

Gerenciamento de Memória

Alocação de segmentos livres

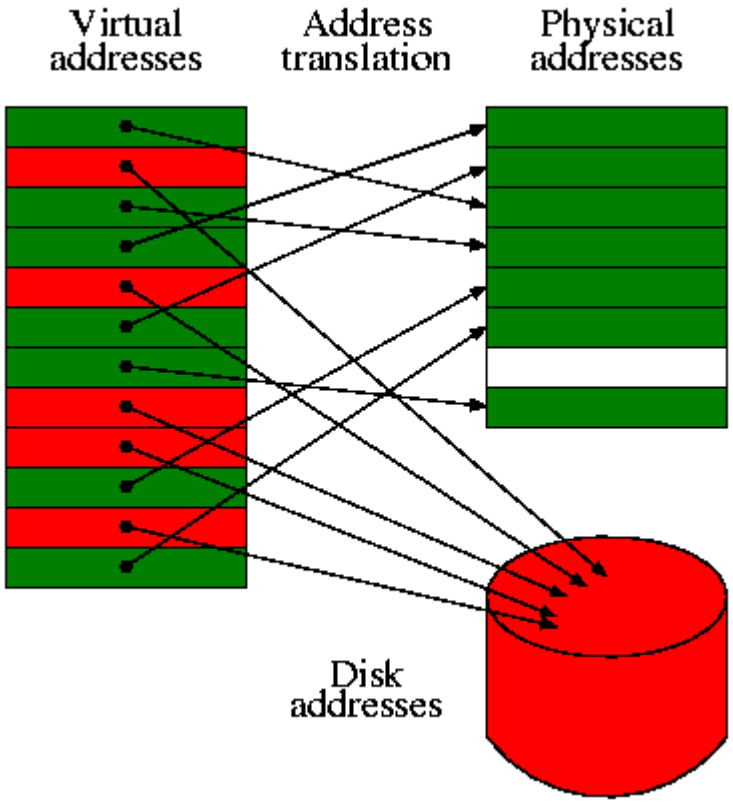
- **Principais Conseqüências**

- **A melhor escolha:** deixa o menor resto, porém após um longo processamento poderá deixar "buracos" muito pequenos para serem úteis.

- **A pior escolha:** deixa o maior espaço após cada alocação, mas tende a espalhar as porções não utilizadas sobre áreas não contínuas de memória e, portanto, pode tornar difícil alocar grandes jobs.

- **A primeira escolha:** tende a ser um meio termo entre a melhor e a pior escolha, com a característica adicional de fazer com que os espaços vazios migrem para o final da memória.

Gerenciamento de Memória
Memória Virtual (MV)
O que é memória virtual?



Gerenciamento de Memória
Memória Virtual (MV)

- Programas maiores que a memória eram divididos em pedaços menores chamados *overlays*;
  - Programador define áreas de *overlay*;
  - Vantagem: expansão da memória principal;
  - Desvantagem: custo muito alto;

Gerenciamento de Memória
Memória Virtual (MV)

- Sistema operacional é responsável por dividir o programa em *overlays*;
- Sistema operacional realiza o chaveamento desses pedaços entre a memória principal e o disco;
- Década de 60: ATLAS → primeiro sistema com MV (Universidade Manchester - Reino Unido);
- 1972: sistema comercial: IBM System/370;

Gerenciamento de Memória
Memória Virtual (MV)

- Com MV existe a sensação de se ter mais memória principal do que realmente se tem;
- O *hardware* muitas vezes implementa funções da gerência de memória virtual:
  - SO deve considerar características da arquitetura;

Gerenciamento de Memória
Memória Virtual

- Espaço de Endereçamento Virtual de um processo é formado por todos os endereços virtuais que esse processo pode gerar;
- Espaço de Endereçamento Físico de um processo é formado por todos os endereços físicos/reais aceitos pela memória principal (RAM);

Gerenciamento de Memória
Memória Virtual

- Um processo em Memória Virtual faz referência a endereços virtuais e não a endereço reais de memória RAM;
- No momento da execução de uma instrução, o endereço virtual é traduzido para um endereço real, pois a CPU manipula apenas endereços reais da memória RAM → MAPEAMENTO;

Gerenciamento de Memória
Memória Virtual

- Técnicas de MV:
  - Paginação:
    - Blocos de tamanho fixo chamados de **páginas**;
    - SO mantém uma fila de todas as páginas;
    - Endereços Virtuais formam o espaço de endereçamento virtual;
    - O espaço de endereçamento virtual é dividido em páginas virtuais;
    - Mapeamento entre endereços reais e virtuais (MMU);

Gerenciamento de Memória
Memória Virtual

- Técnicas de MV:
  - Segmentação:
    - Blocos de tamanho arbitrário chamados **segmentos**;
- Arquitetura (hardware) tem que possibilitar a implementação tanto da paginação quanto da segmentação;

Gerenciamento de Memória
Memória Virtual - *Swapping*

- *Swapping*: chaveamento de processos inteiros entre a memória principal e o disco;
  - *Swap-out*;
  - *Swap-in*;
  - Pode ser utilizado tanto com partições fixas quanto com partições variáveis;

Gerenciamento de Memória
Memória Virtual - Paginação

- Memória Principal e Memória Secundária são organizadas em páginas de mesmo tamanho;
- Página é a unidade básica para transferência de informação;
- Tabela de páginas: responsável por armazenar informações sobre as páginas virtuais:
  - argumento de entrada → número da página virtual;
  - argumento de saída (resultado) → número da página real (ou moldura de página - *page frame*);

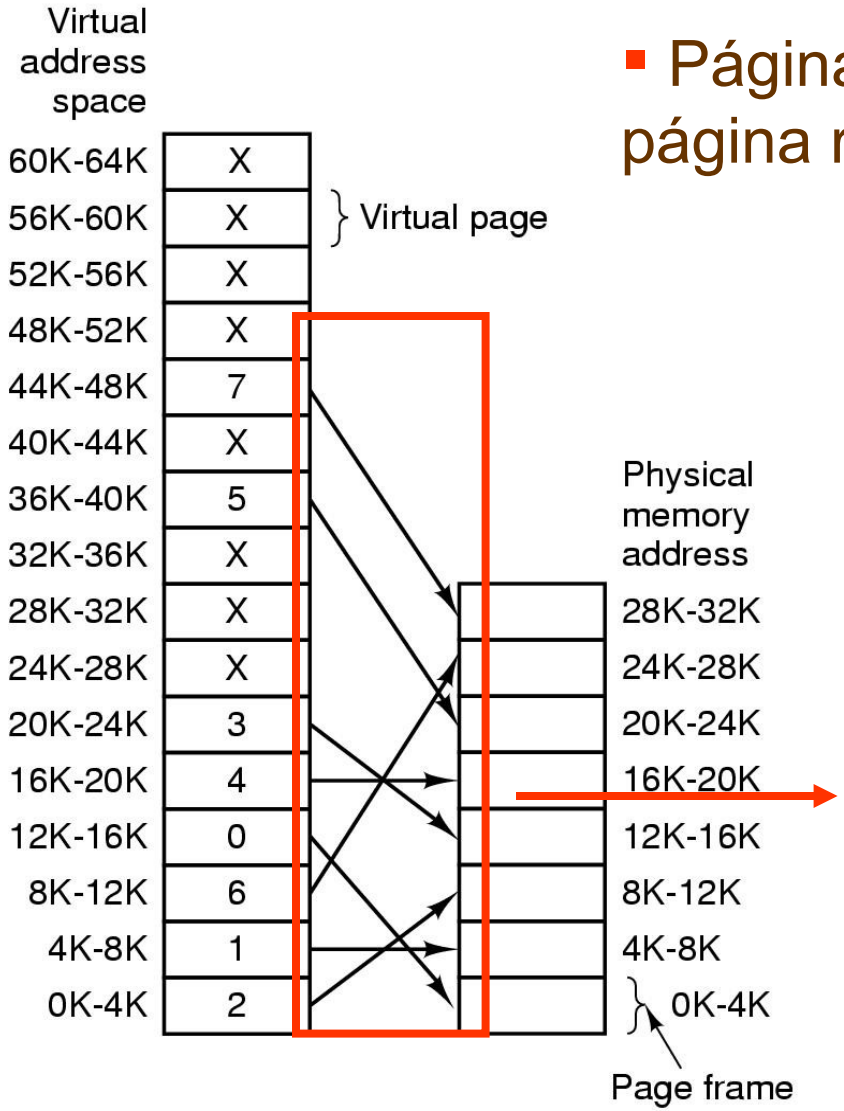
Gerenciamento de Memória
Memória Virtual

- Exemplo:
  - Páginas de 4Kb
    - 4096 bytes/endereços (0-4095);
  - 64Kb de espaço virtual;
  - 32Kb de espaço real;
  - Temos:
    - 16 páginas virtuais;
    - 8 páginas reais;

Gerenciamento de Memória
Memória Virtual - Paginação

- Problemas:
  - Fragmentação interna;
  - Definição do tamanho das páginas;
    - Geralmente a MMU que define e não o SO;
    - Páginas maiores: leitura mais eficiente, tabela menor, mas maior fragmentação interna;
    - Páginas menores: leitura menos eficiente, tabela maior, mas menor fragmentação interna;
    - Sugestão: 1k a 8k;
- Mapa de bits ou uma lista encadeada com as páginas livres;

Gerenciamento de Memória
Endereço Virtual → Endereço Real

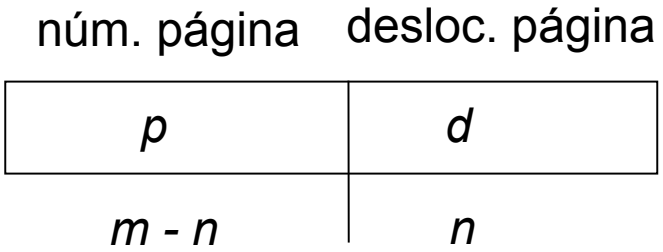


- Página virtual **mapeada** para página real;

- MMU realiza o mapeamento

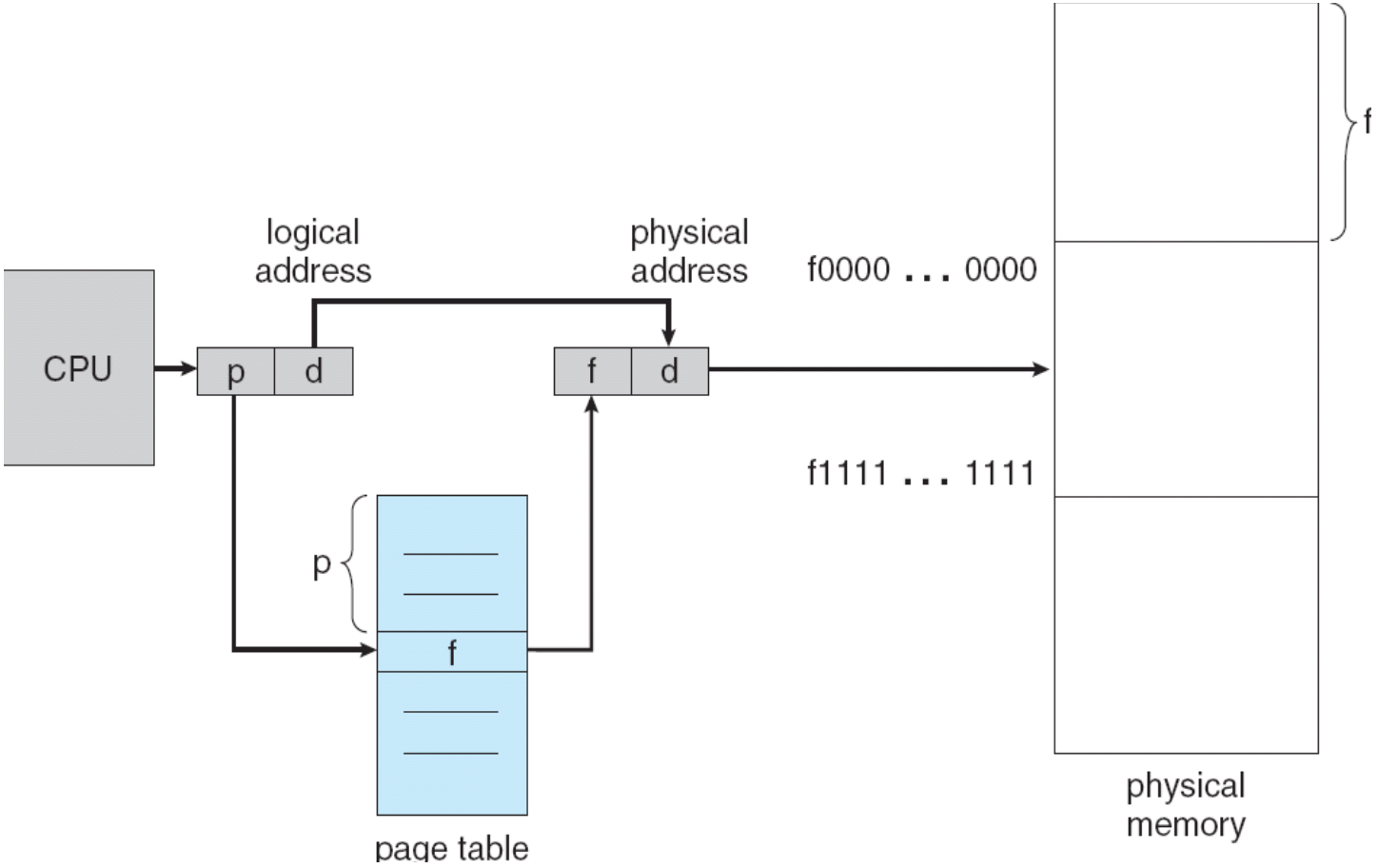
Esquema de tradução de endereço

- 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

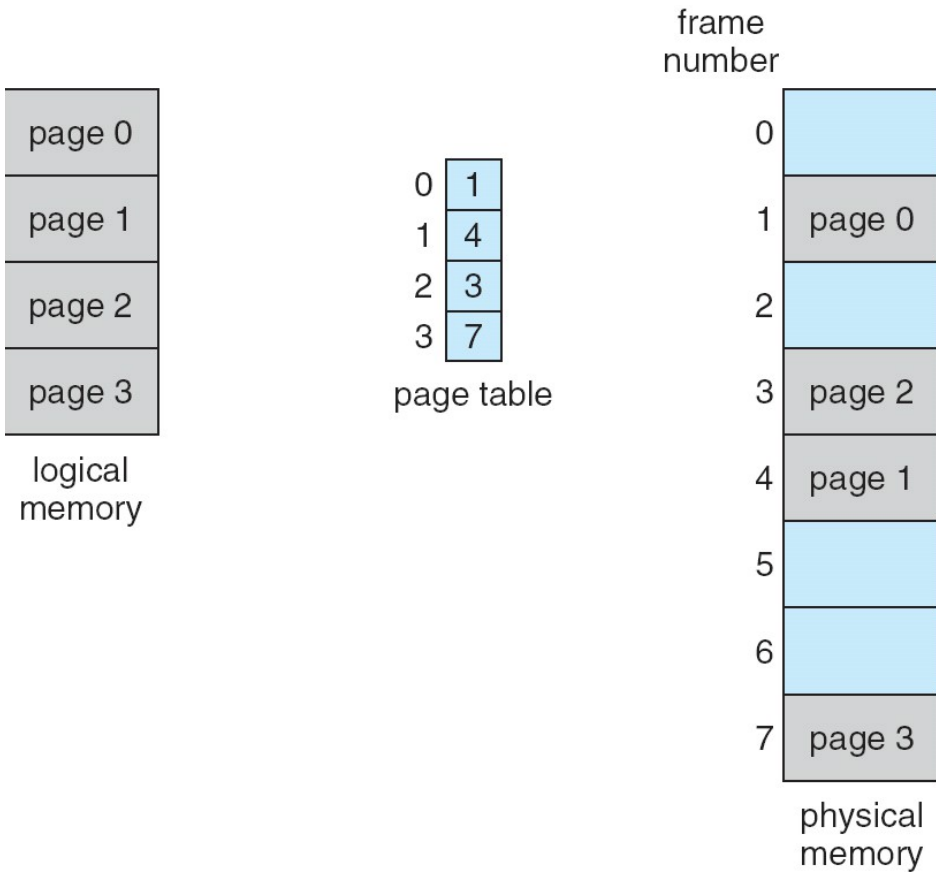


  - Para determinado espaço de endereço lógico 2^m e tamanho de página 2^n

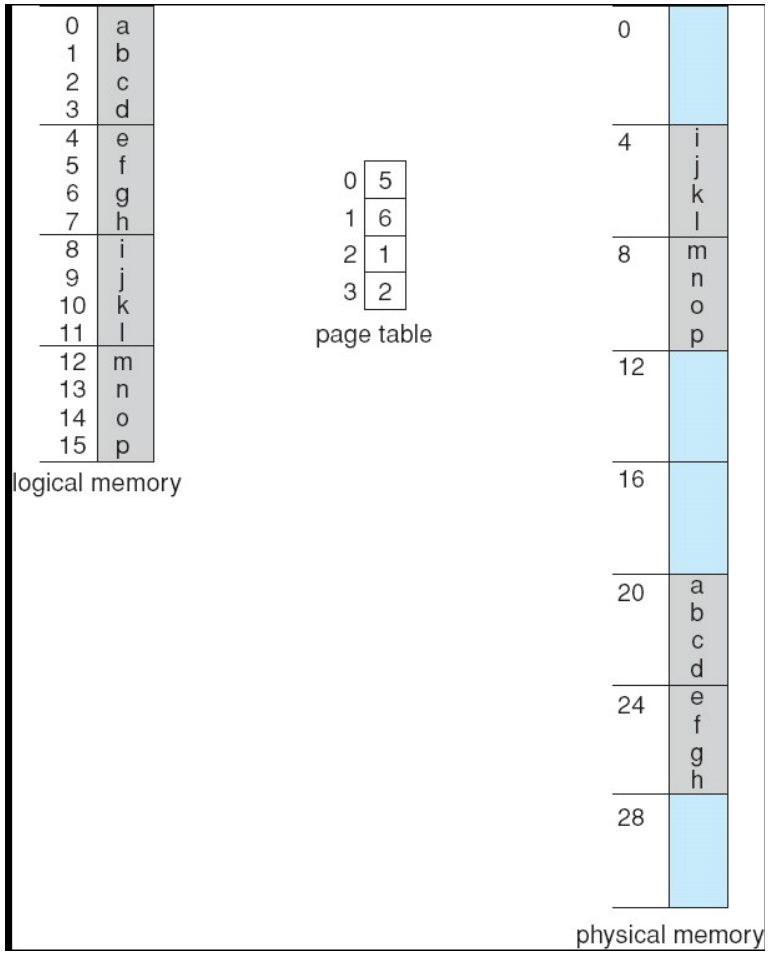
Hardware de paginação



Modelo de paginação da memória lógica e física



Exemplo de paginação



memória de 32 bytes e páginas de 4 bytes

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

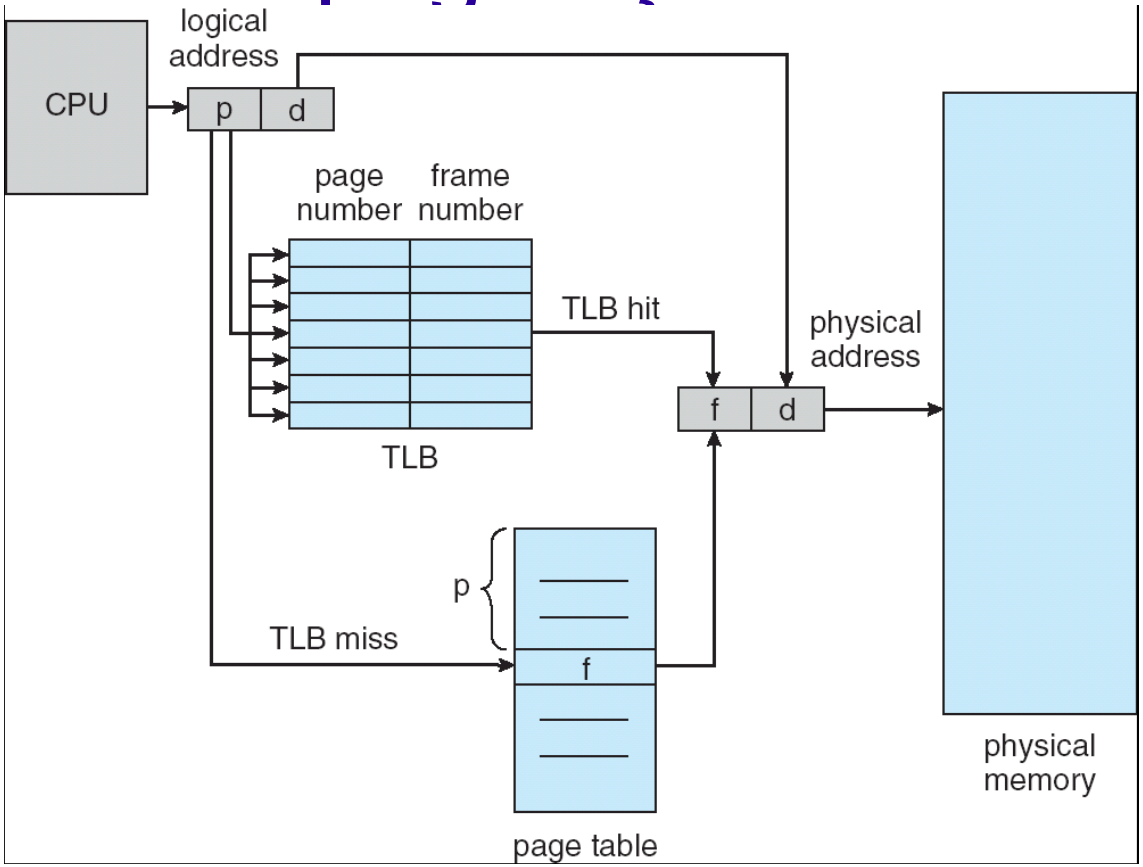
TLB ou Memória associativa

- Memória associativa – busca paralela

Pág. #	Quadro #

* TLBs e loops;
* Tradução de endereço (p, d)
  - Se p está no registrador associativo, retira quadro #
  - Caso contrário, coloca quadro # da tabela de página para a memória associativa

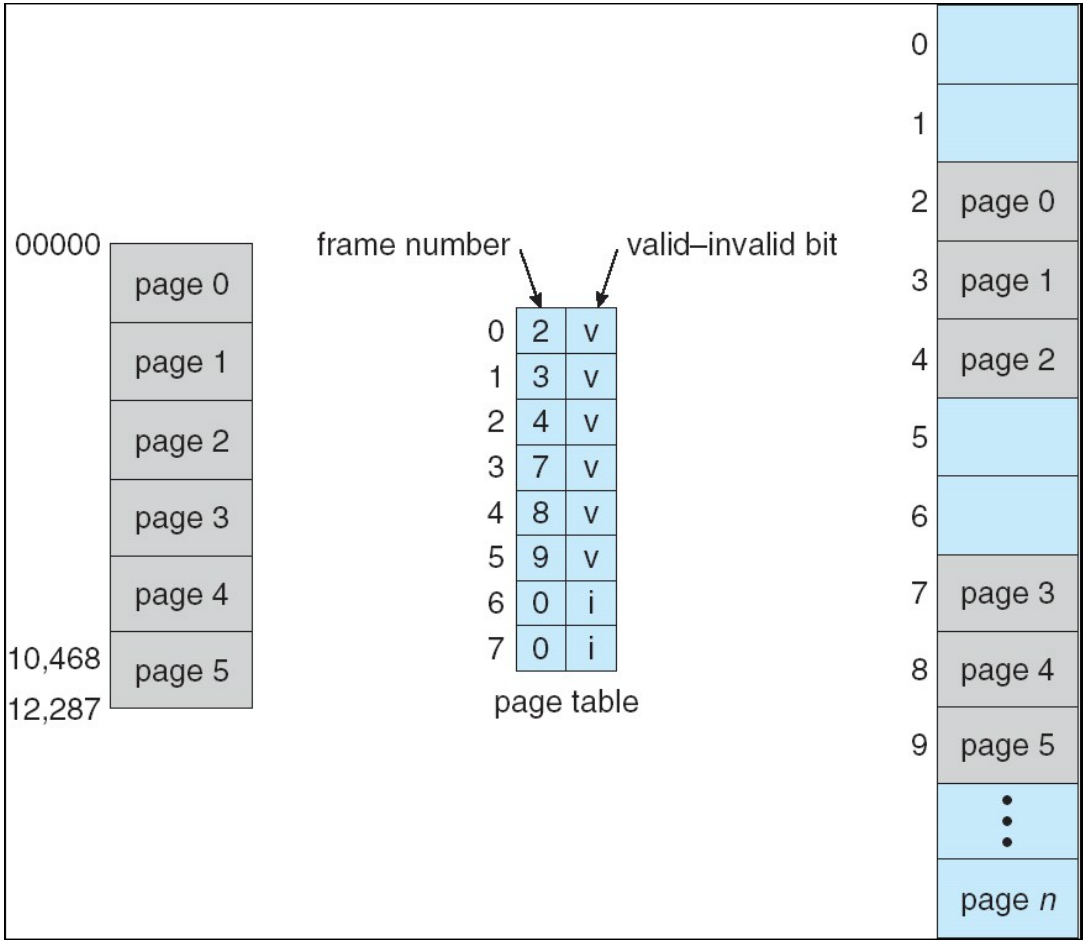
Hardware de paginação com TLB



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 (v) ou inválido (i) em uma tabela de página

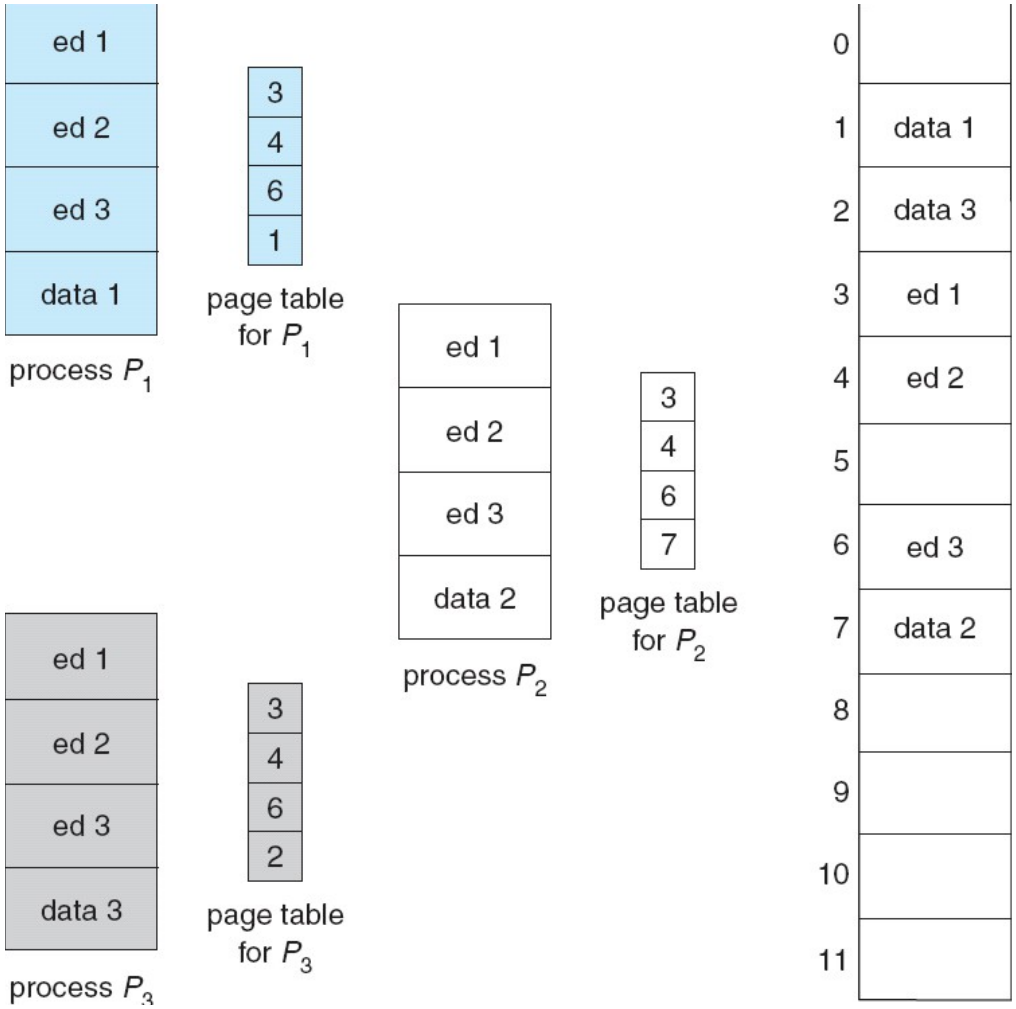


Páginas compartilhadas

- **Código compartilhado**
  - Uma cópia de código somente de leitura (reentrante) 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



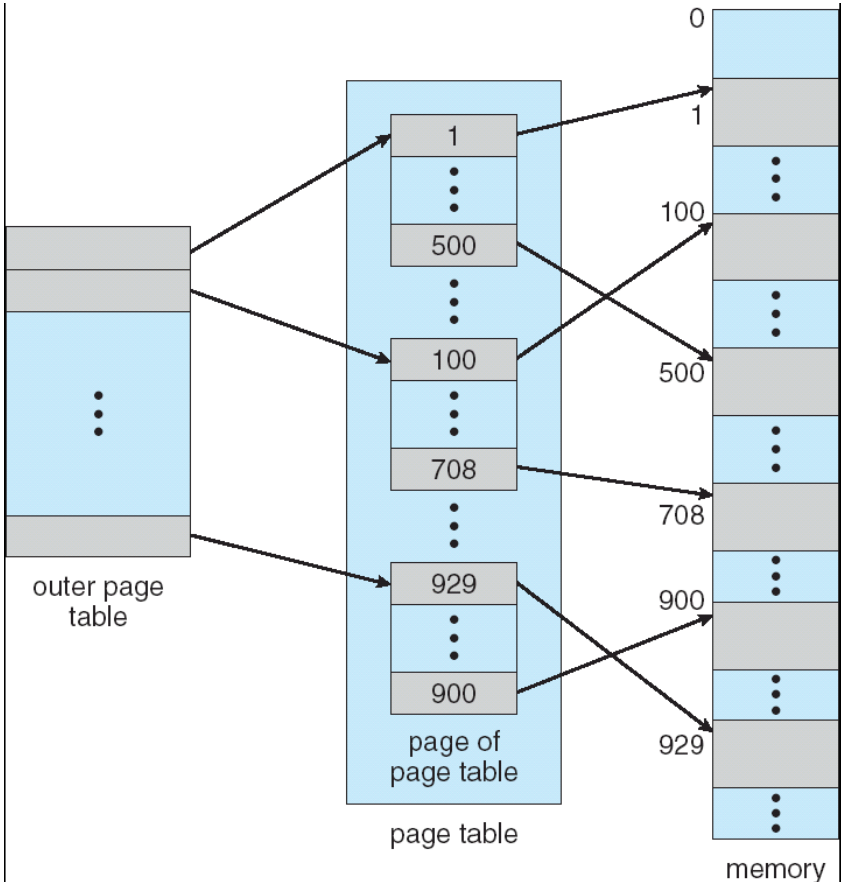
Estrutura da tabela de página

- Como a tabela de páginas é organizada?
  - Paginação hierárquica
  - Tabelas de página com hash
  - Tabelas de página invertidas

Tabelas de página hierárquicas

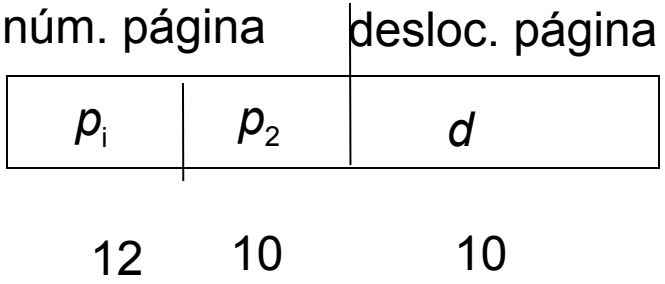
- Quebre o espaço de endereço lógico em múltiplas tabelas de página
- Uma técnica simples é uma tabela de página em dois níveis

Esquema de tabela de página em dois níveis



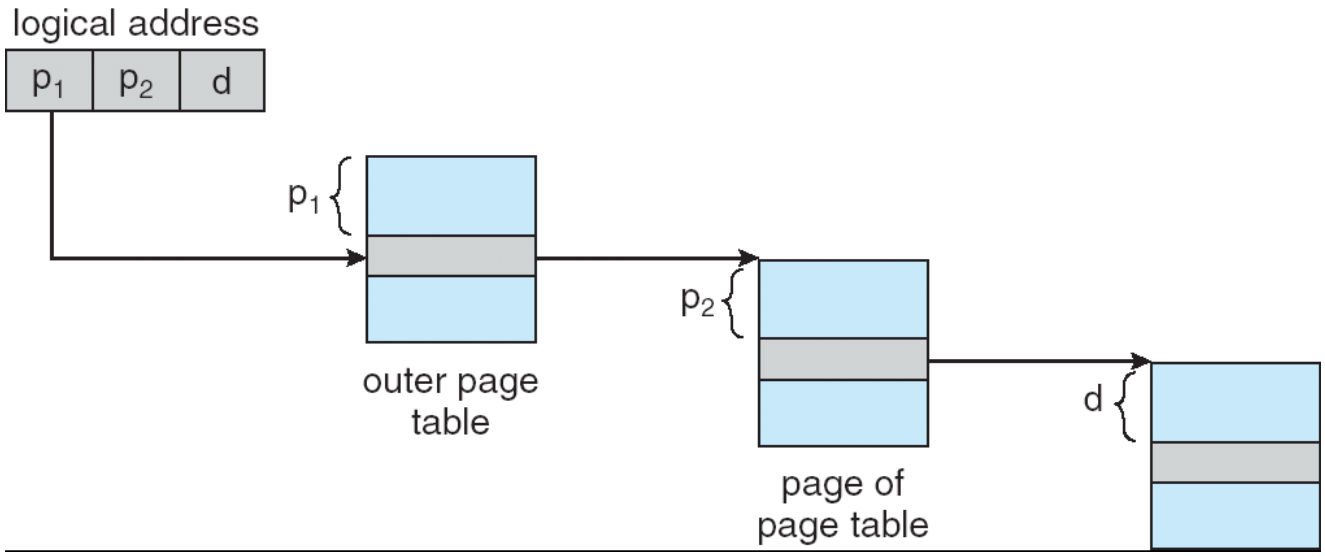
Exemplo de paginação em dois níveis

- 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

- Comuns em espaços de endereço > 32 bits
- O número de página virtual é dividido em uma tabela de página. Essa tabela de página consiste em uma cadeia de elementos que se traduzem para o mesmo local.
- Números de página virtual são comparados nessa cadeia buscando uma combinação. Se uma combinação for achada, o quadro físico correspondente é extraído.

Tabela de página em hash

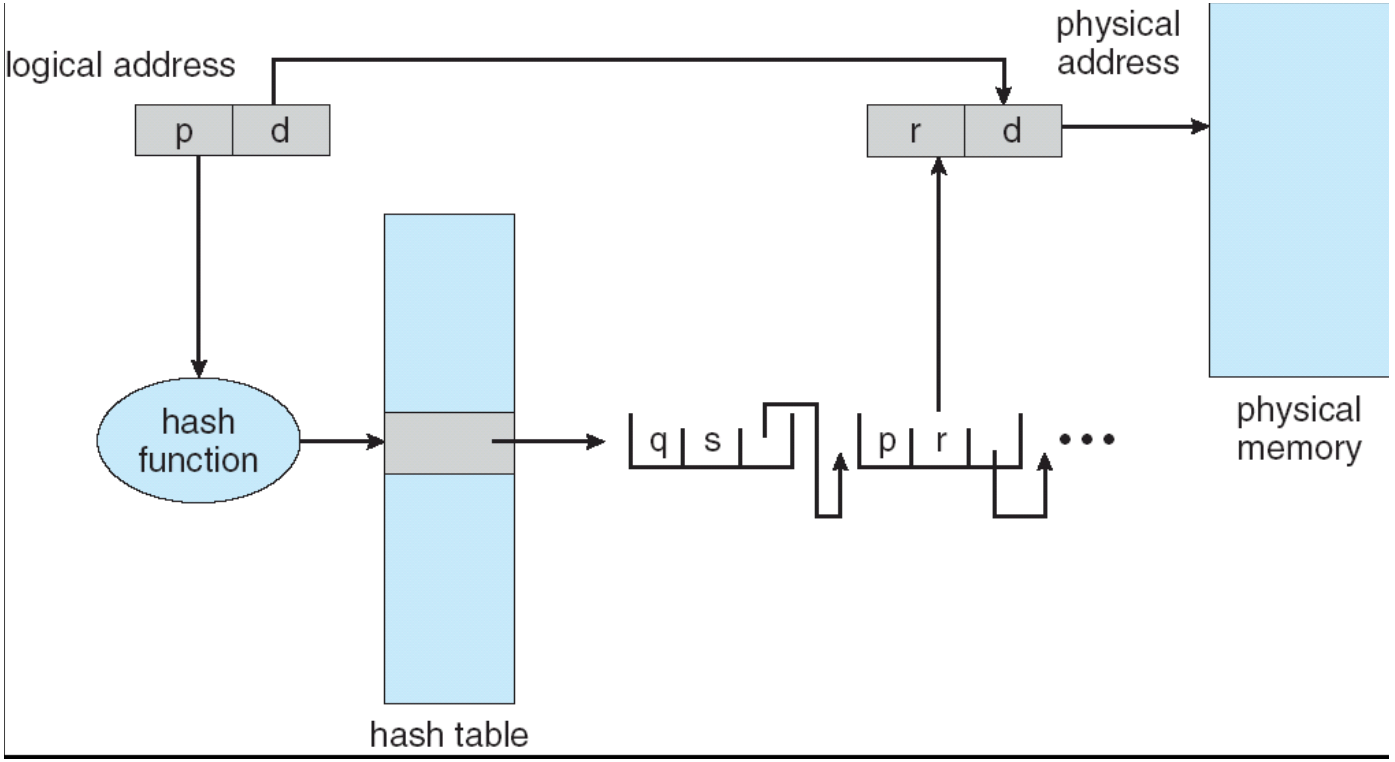
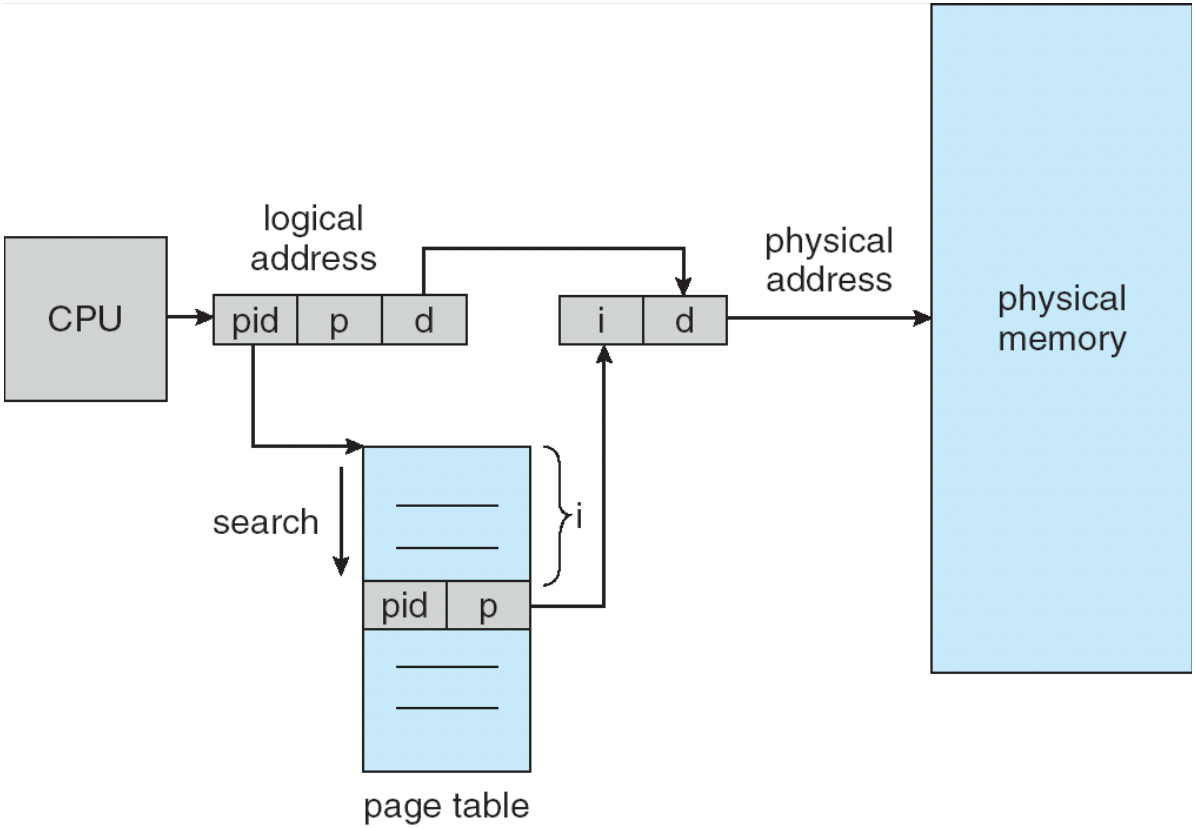


Tabela de página invertida

- 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, 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

- Tabela ordenada pela memória física
- pid é o número do processo



Gerenciamento de Memória
Tabela de Páginas Invertida

- Geralmente, cada processo tem uma tabela de páginas associada a ele → classificação feita pelo endereço virtual;
  - Pode consumir grande quantidade de memória;
- Alternativa: tabela de páginas invertida;
  - SO mantém uma única tabela para as molduras de páginas da memória;
  - Cada entrada consiste no endereço virtual da página armazenada naquela página real, com informações sobre o processo dono da página virtual;
  - Exemplos de sistemas: IBM System/38, IBM RISC System 6000, IBM RT e estações HP Spectrum;

Gerenciamento de Memória
Tabela de Páginas Invertida

- Quando uma referência de memória é realizada (página virtual), a tabela de páginas invertida é pesquisada para encontrar a moldura de página correspondente;
  - Se encontra, o endereço físico é gerado → <i, deslocamento>;

Gerenciamento de Memória
Tabela de Páginas Invertida

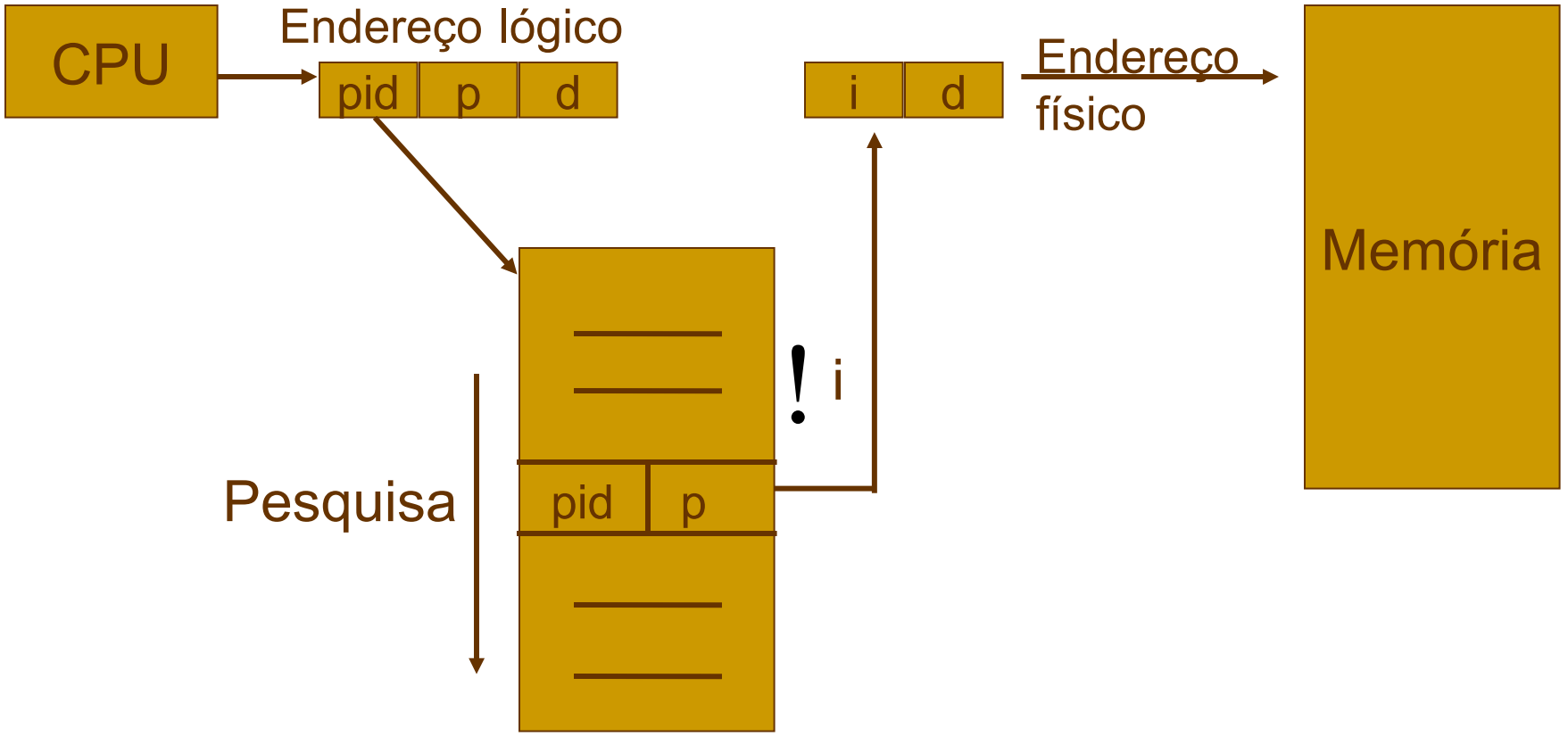


Tabela de páginas invertida

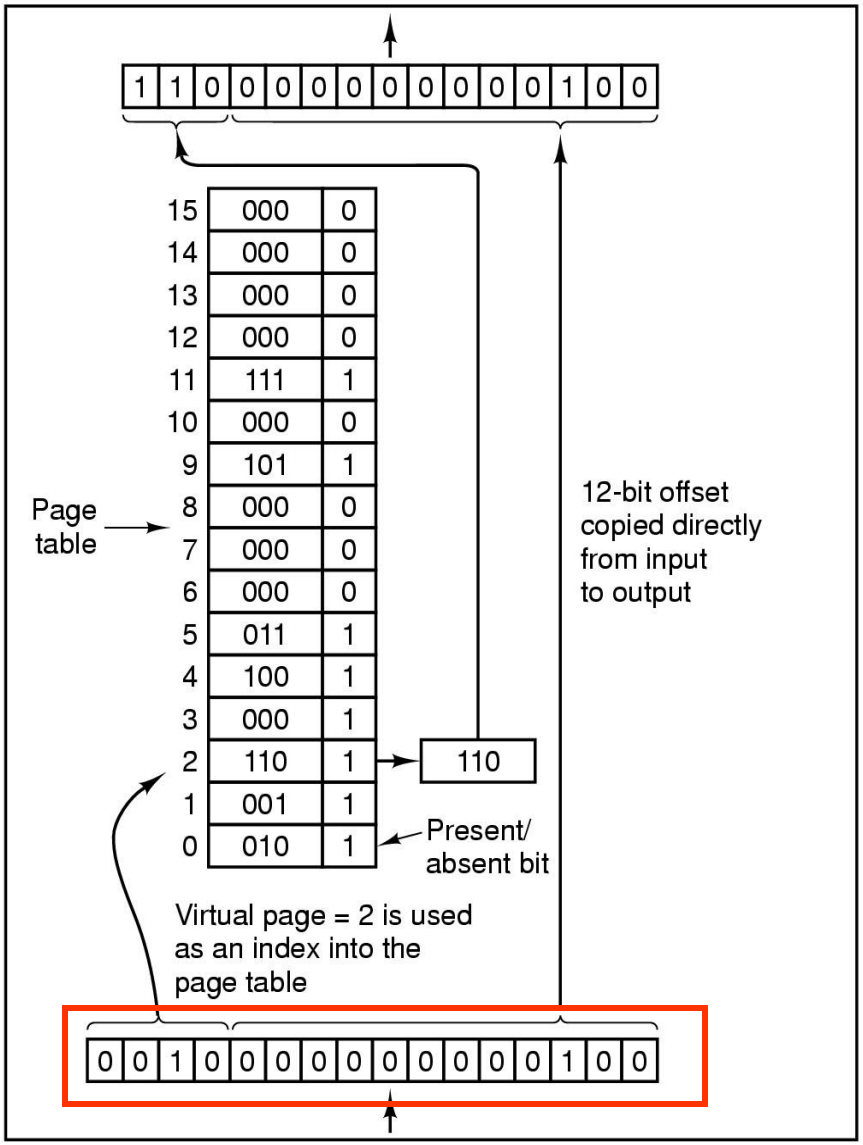
Endereço lógico: <id processo (pid), número página (p), deslocamento (d)>

Gerenciamento de Memória
Tabela de Páginas Invertida

- Vantagens:
  - Ocupa menos espaço;
  - É mais fácil de gerenciar apenas uma tabela;
- Desvantagens:
  - Aumenta tempo de pesquisa na tabela, pois, apesar de ser **classificada por endereços físicos,** é **pesquisada por endereços lógicos**;
  - Aliviar o problema: tabela de *hashing*;
    - Uso da TLB (memória associativa) para manter entradas recentemente utilizadas;

Gerenciamento de Memória

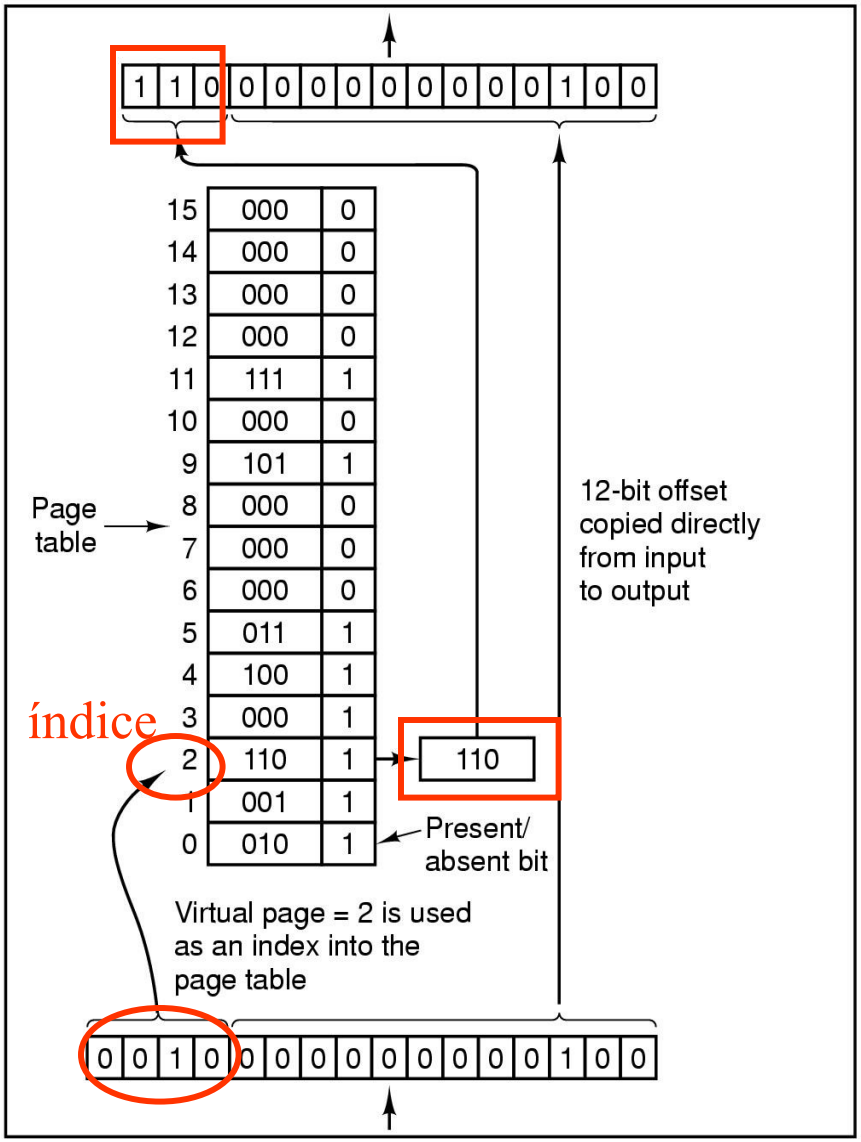
Mapeamento da MMU



- Operação interna de uma MMU com 16 páginas de 4Kb;
- Endereço virtual de **16 bits**: 4 bits para nº de páginas e 12 para deslocamento;
- Com 4 bits é possível ter 16 páginas virtuais (2^4);
- 12 bits para deslocamento
é possível endereçar os 4096 bytes;

Gerenciamento de Memória

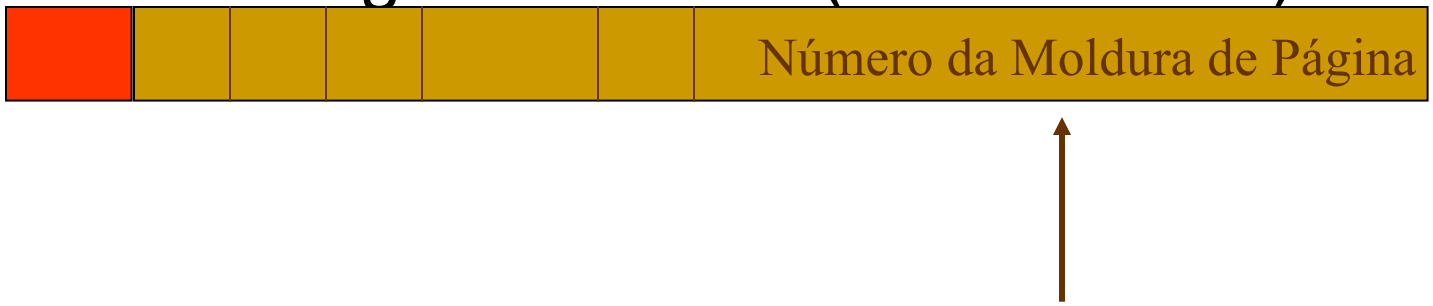
Mapeamento da MMU



- Número da página virtual é usado como índice;
- Se página está na memória RAM, então o nº da página real (110) é copiado para os três bits mais significativos do endereço de saída (real), juntamente com o deslocamento sem alteração;
- Endereço real com 15 bits é enviado à memória;

Gerenciamento de Memória
Memória Virtual - Paginação

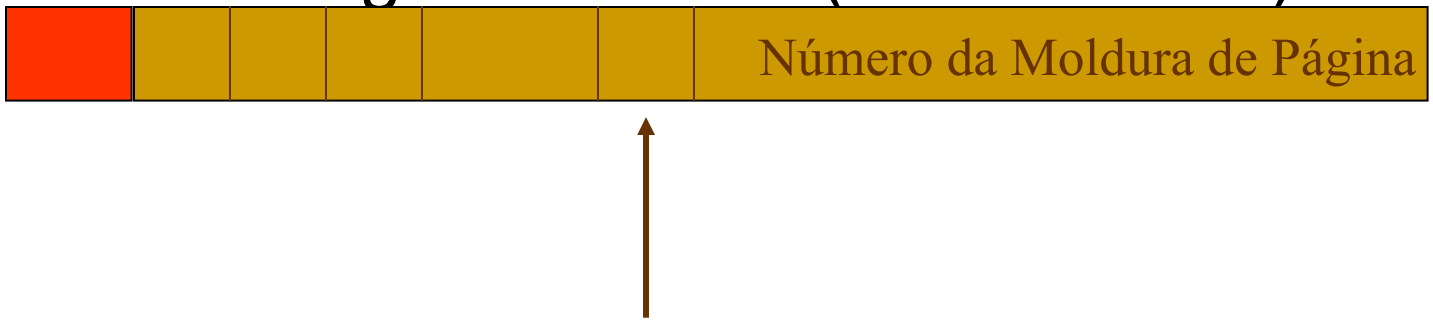
- Tabela de Páginas: 32 bits (mais comum)



Identifica a página real;
Campo mais importante;

Gerenciamento de Memória
Memória Virtual - Paginação

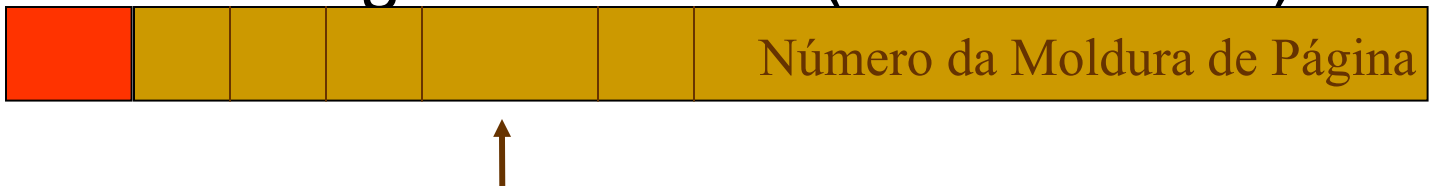
- Tabela de Páginas: 32 bits (mais comum)



***Bit* de Residência:**
Se valor igual 1, então entrada válida para uso;
Se valor igual 0, então entrada inválida, pois página virtual correspondente não está na memória;

Gerenciamento de Memória
Memória Virtual - Paginação

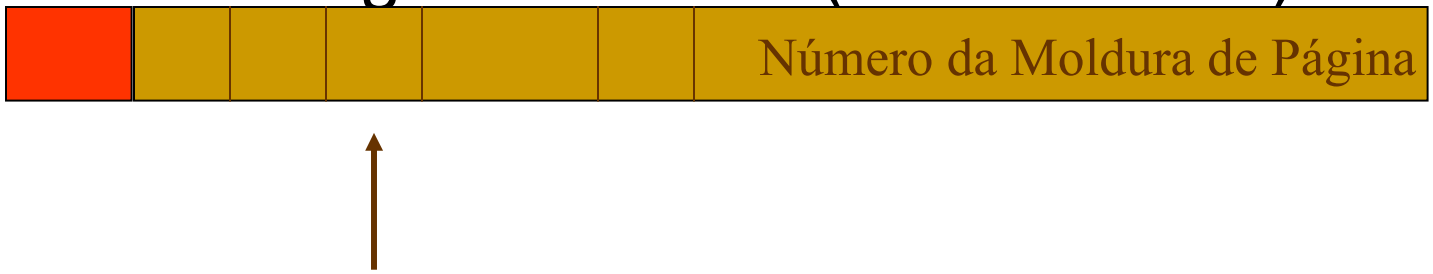
- Tabela de Páginas: 32 bits (mais comum)



***Bits* de Proteção:**
Indicam tipos de acessos permitidos:
1 bit → 0 – leitura/escrita
1 – leitura
3 bits → 0 – Leitura
1 – Escrita
2 - Execução

Gerenciamento de Memória
Memória Virtual - Paginação

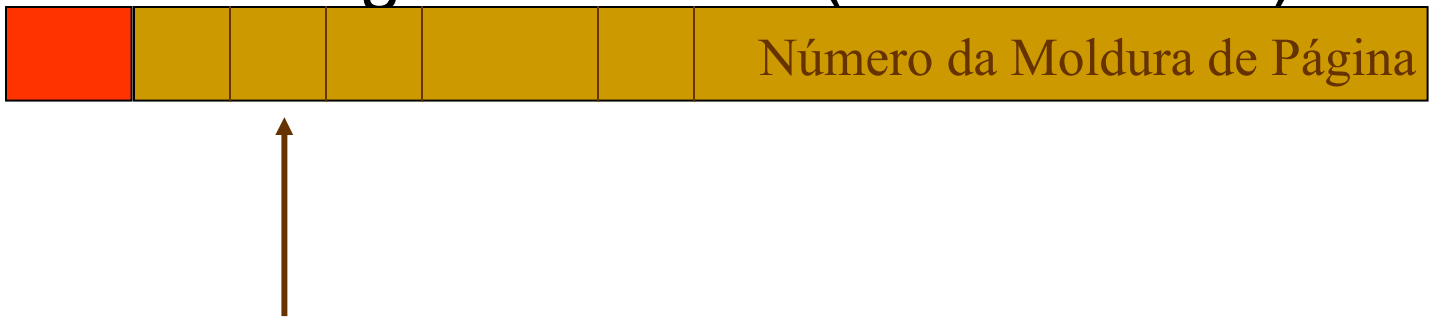
- Tabela de Páginas: 32 bits (mais comum)



***Bit* de Modificação (Bit M):**
Controla o uso da página;
Se valor igual a 1, página foi escrita;
página é copiada para o disco
Se valor igual a 0, página não foi modificada;
página não é copiada para o disco;

Gerenciamento de Memória
Memória Virtual - Paginação

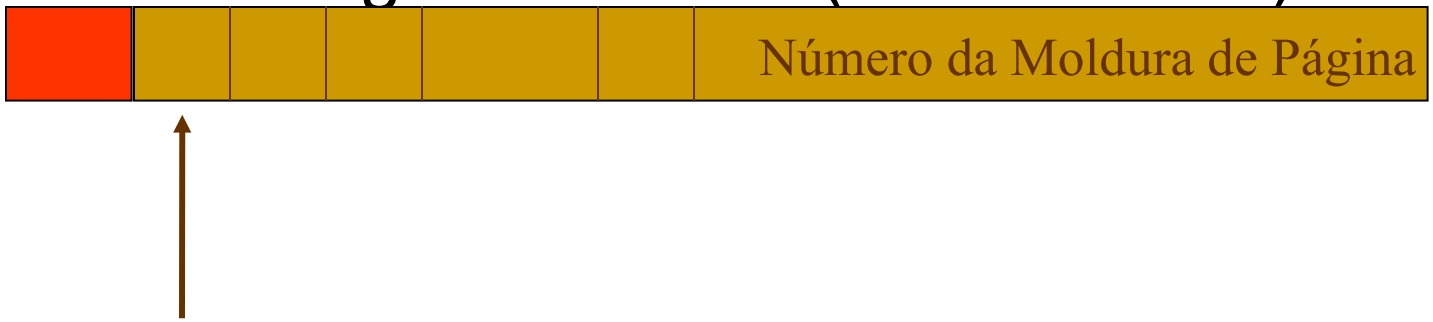
- Tabela de Páginas: 32 bits (mais comum)



***Bit* de Referência (Bit R):**
Controla o uso da página;
Auxilia o SO na escolha da página que deve deixar a MP (RAM);
Se valor igual a 1, página foi referenciada (leitura/escrita);
Se valor igual a 0, página não referenciada;

Gerenciamento de Memória
Memória Virtual - Paginação

- Tabela de Páginas: 32 bits (mais comum)



***Bit* de *Cache*:**
Necessário quando os dispositivos de entrada/saída são mapeados na memória e não em um endereçamento específico de E/S;

Gerenciamento de Memória
Memória Virtual - Paginação

- A Tabela de páginas pode ser armazenada de três diferentes maneiras:
  - Registradores, se a memória for pequena;
  - Na própria memória RAM → MMU gerencia utilizando dois registradores:
    - Registrador Base da tabela de páginas (PTBR – *page table base register*): indica o endereço físico de memória onde a tabela está alocada;
    - Registrador Limite da tabela de páginas (PTLR – *page table limit register*): indica o número de entradas da tabela (número de páginas);
    - Dois acessos à memória (um para a tabela e outra para a RAM);

Gerenciamento de Memória
Memória Virtual - Paginação

- Em uma memória *cache* na MMU chamada Memória Associativa (TLB);
  - Também conhecida como TLB (*Translation Lookaside Buffer - buffer* de tradução dinâmica*)*;
  - Hardware especial para mapear endereços virtuais para endereços reais sem ter que passar pela tabela de páginas na memória principal;
  - Funciona como a cache das páginas virtuais (páginas mais acessadas)

Até 32/64 e não mais do que 256 entradas

Bit R	Página Virtual	*Bit* M	*Bits* de Proteção	Página Física
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45

Gerenciamento de Memória
Alocação de Páginas

- Quantas páginas reais serão alocadas a um processo;
- Duas estratégias:
  - Alocação fixa ou estática: cada processo tem um número máximo de páginas reais, definido quando o processo é criado;
    - O limite pode ser igual para todos os processos;
    - Vantagem: simplicidade;
    - Desvantagens: (i) número muito pequeno de páginas reais pode causar muita paginação; (ii) número muito grande de páginas reais causa desperdício de memória principal;

Gerenciamento de Memória
Alocação de Páginas

- Alocação variável ou dinâmica: número máximo de páginas reais alocadas ao processo varia durante sua execução;
  - Vantagens:
    - processos com elevada taxa de paginação podem ter seu limite de páginas reais ampliado;
    - processos com baixa taxa de paginação podem ter seu limite de páginas reais reduzido;
  - Desvantagem: monitoramente constante;

Gerenciamento de Memória
Busca de Página

- **Paginação simples**:
  - Todas as páginas virtuais do processo são carregadas para a memória principal;
  - Assim, sempre todas as páginas são válidas;
- **Paginação por demanda** (*Demand Paging*):
  - Apenas as páginas efetivamente acessadas pelo processo são carregadas na memória principal;
  - Quais páginas virtuais foram carregadas → Bit de controle (bit de residência);
  - Página inválida;

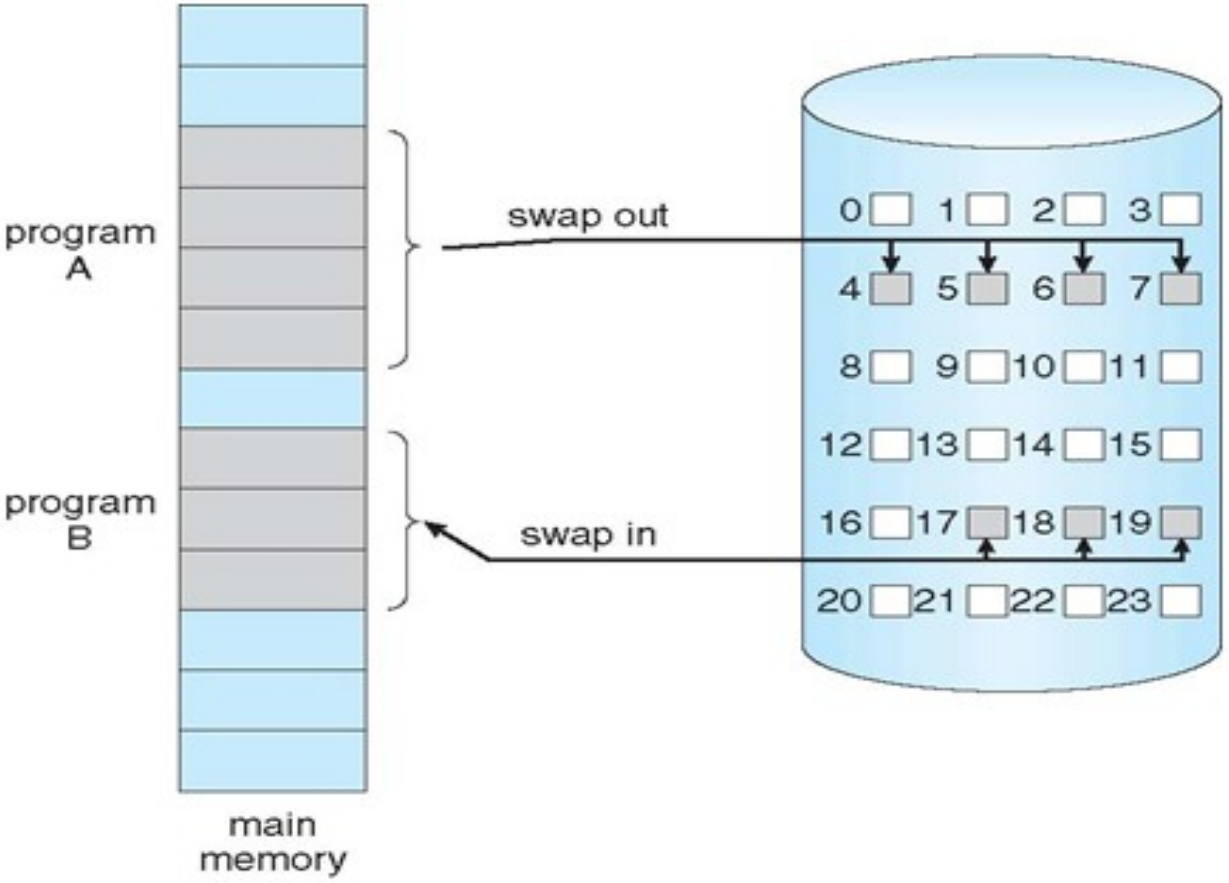
Gerenciamento de Memória
Busca de Página

- Página inválida: MMU gera uma interrupção de proteção e aciona o sistema operacional;
  - Se a página está fora do espaço de endereçamento do processo, o processo é abortado;
  - Se a página ainda não foi carregada na memória principal, ocorre uma **falta de página** *(page fault);*

Gerenciamento de Memória
Busca de Página

- **Falta de Página**:
  - Processo é suspenso e seu descritor é inserido em uma **fila especial** – fila dos processos esperando uma página virtual;
  - Uma página real livre deve ser alocada;
  - A página virtual acessada deve ser localizada no disco;
  - Operação de leitura de disco, indicando o endereço da página virtual no disco e o endereço da página real alocada;

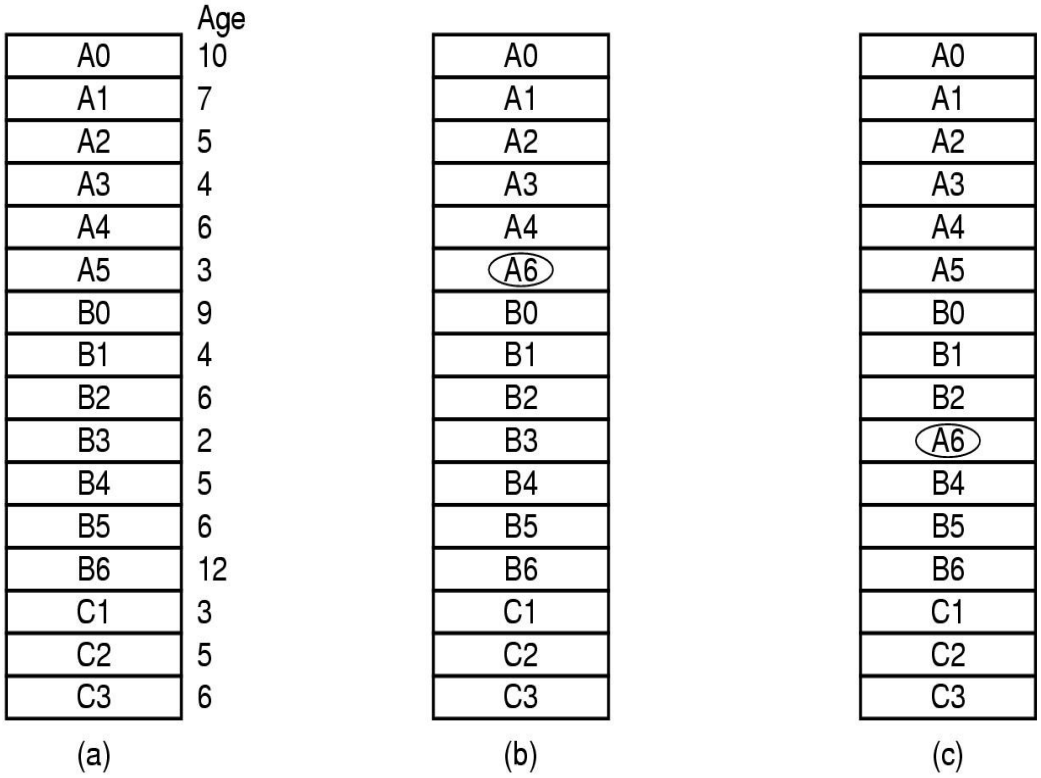
Gerenciamento de Memória
Busca de Página



Gerenciamento de Memória
Troca de Páginas

- Política de Substituição Local: páginas dos próprios processos são utilizadas na troca;
  - Dificuldade: definir quantas páginas cada processo pode utilizar;
- Política de Substituição Global: páginas de todos os processos são utilizadas na troca;
  - Problema: processos com menor prioridade podem ter um número muito reduzido de páginas, e com isso, acontecem muitas faltas de páginas;

Gerenciamento de Memória
Troca de Páginas



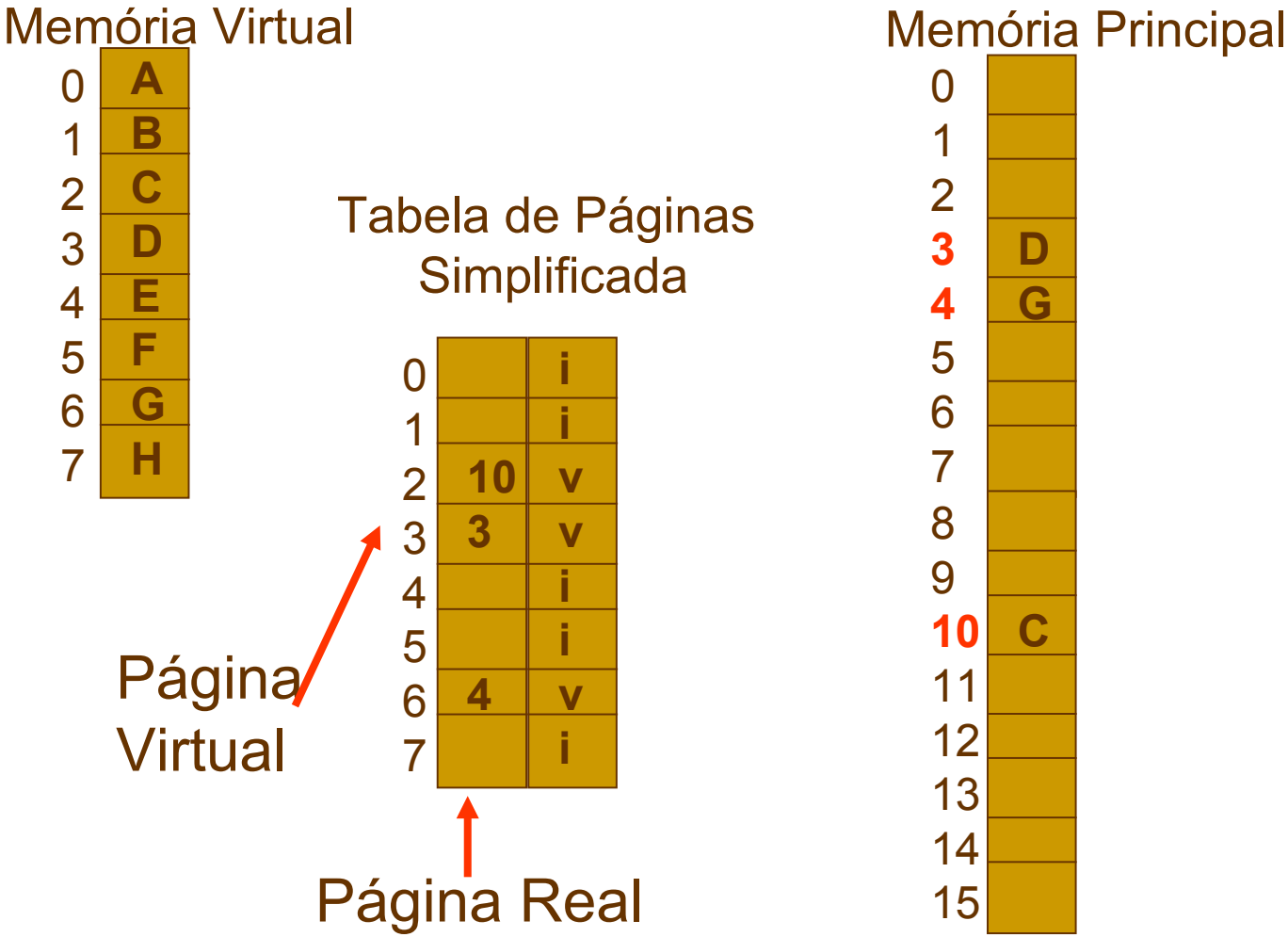
- a) Configuração inicial;
- b) Alocação local;
- c) Alocação global;

Gerenciamento de Memória
Troca de Páginas

- Algoritmos de substituição local alocam uma fração fixa de memória para cada processo;

- Algoritmos de substituição global alocam molduras de páginas entre os processos em execução
  - Resultado: há a variação do # de quadros de páginas no tempo por processo

Gerenciamento de Memória
Troca de Páginas



Gerenciamento de Memória
Troca de Páginas

- Se todas as páginas estiverem ocupadas, uma página deve ser retirada: página vítima;
- Exemplo:
  - Dois processos P1 e P2, cada um com 4 páginas virtuais;
  - Memória principal com 6 páginas;

Gerenciamento de Memória
Troca de Páginas (8 pág. vs. 6 frames)

Memória Virtual P1

0	A
1	B
2	C
3	D

Tabela de Páginas P1 Simplificada

0	1	v
1	5	v
2		i
3	0	v

Memória Principal

0	D
1	A
2	F
3	E
4	G
5	B

3 páginas de cada processo

Memória Virtual P2

0	E
1	F
2	G
3	H

Tabela de Páginas P2 Simplificada

0	3	v
1	2	v
2	4	v
3		i

→ P2 tenta acessar página 3! Falta de Página!

Gerenciamento de Memória
Troca de Páginas

Memória Virtual P1

0	A
1	B
2	C
3	D

Tabela de Páginas P1 Simplificada

0	1	v
1	5	v
2		i
3	0	v

Memória Principal

0	D
1	A
2	F
3	E
4	H
5	B

3 páginas de cada processo
JUSTIÇA

Memória Virtual P2

0	E
1	F
2	G
3	H

Tabela de Páginas P2 Simplificada

0	3	v
1	2	v
2		i
3	4	v

→ Página 2 (virtual) é escolhida como vítima!

Gerenciamento de Memória
Troca de Páginas - Paginação

- Algoritmos:
  - Ótimo;
  - NRU;
  - FIFO;
  - Segunda Chance;
  - Relógio;
  - LRU;
  - *Working set*;
  - *WSClock;*

Gerenciamento de Memória
Troca de Páginas - Paginação

- Algoritmo Ótimo:
  - Retira da memória a página que tem menos chance de ser referenciada;
    - Praticamente impossível de se saber;
    - Impraticável;
    - Usado em simulações para comparação com outros algoritmos;

Gerenciamento de Memória
Troca de Páginas - Paginação

- Algoritmo *Not Recently Used Page Replacement* (NRU) → troca as páginas não utilizadas recentemente:
  - 02 bits associados a cada página → R e M
    - Classe 0 → não referenciada, não modificada;
    - Classe 1 → não referenciada, modificada (do tipo 3 que não foi referenciada porque o clock limpa periodicamente);
    - Classe 2 → referenciada, não modificada;
    - Classe 3 → referenciada, modificada;
  - R e M são atualizados a cada referência à memória;

Gerenciamento de Memória
Troca de Páginas - Paginação

- NRU:
  - Periodicamente, o bit R é limpo para diferenciar as páginas que não foram referenciadas recentemente;
    - A cada *tick* do relógio ou interrupção de relógio;
    - Classe 3 → Classe 1;
  - Vantagens: fácil de entender, eficiente para implementar e fornece bom desempenho;

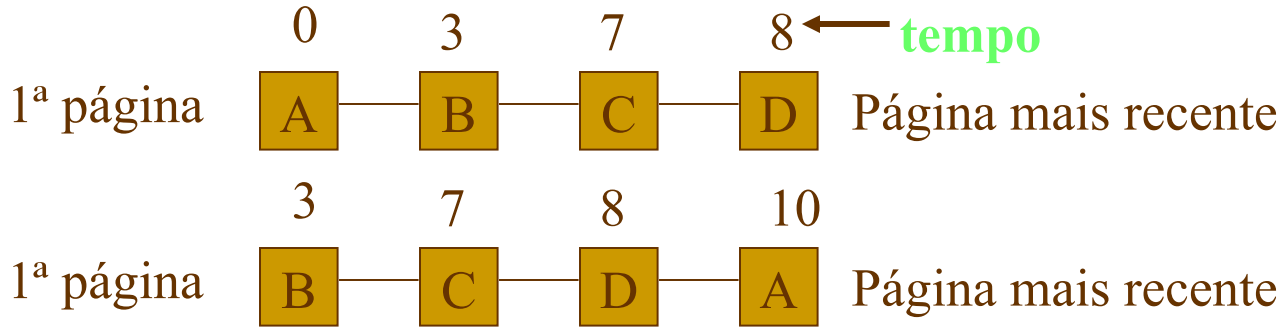
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *First-in First-out Page Replacement* (FIFO)
  - SO mantém uma lista das páginas correntes na memória;
    - A página no início da lista é a mais antiga e a página no final da lista é a mais nova;
  - Simples, mas pode ser ineficiente, pois uma página que está em uso constante pode ser retirada;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo da Segunda Chance
  - FIFO + *bit* R;
  - Página mais velha é candidata em potencial;

Se o bit R==0, então página é retirada da memória,
senão, R=0
e se dá uma nova chance à página colocando-a no final da lista;



Se página A com R==1; e falta de página em tempo 10; Então R=0 e página A vai para final da lista;

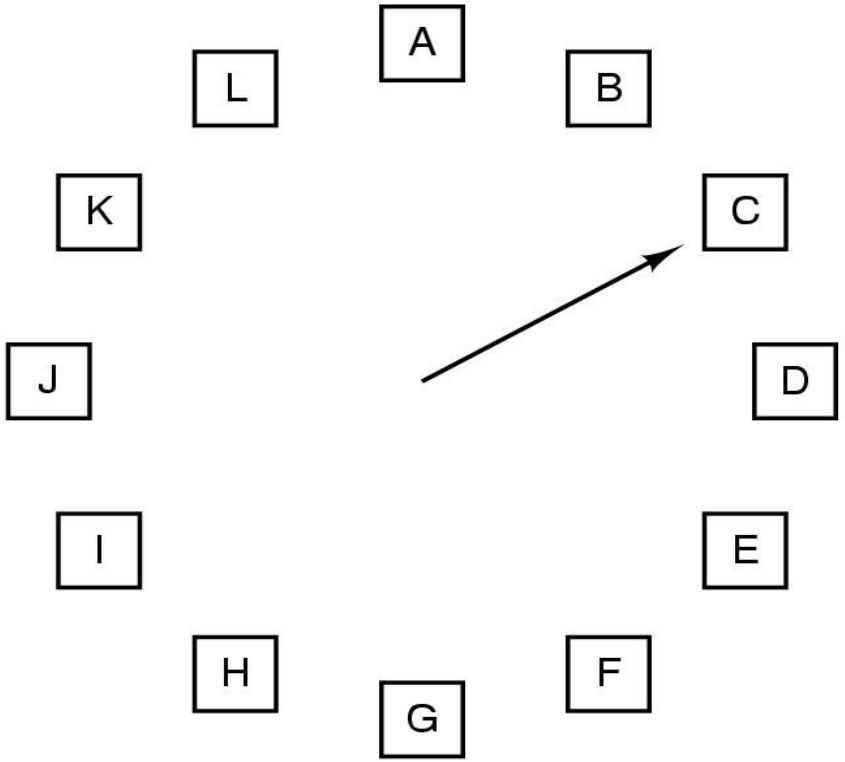
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo do Relógio
  - Lista circular com ponteiro apontando para a página mais antiga
  - Algoritmo se repete até encontrar R=0;

Se R=0	Se R=1
- troca de página - desloca o ponteiro	- R = 0 - desloca o ponteiro - continua busca

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo do Relógio



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Least Recently Used Page Replacement* (LRU)
  - Troca a página menos referenciada/modificada recentemente;
  - Alto custo
    - Lista encadeada com as páginas que estão na memória, com as mais recentemente utilizadas no início e as menos utilizadas no final;
    - A lista deve ser atualizada a cada referência da memória;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Least Recently Used Page Replacement* (LRU)
  - Pode ser implementado tanto por hardware quanto por software:
    - Hardware: MMU deve suportar a implementação LRU;
      - Contador em hardware (64 *bits*);
      - Tabela de páginas armazena o valor desse contador para saber quantas vezes a página foi usada;
    - Software: duas maneiras
      - NFU (*Not frequently used*);
      - *Aging* (Envelhecimento);

Gerenciamento de Memória
Troca de Páginas - Paginação

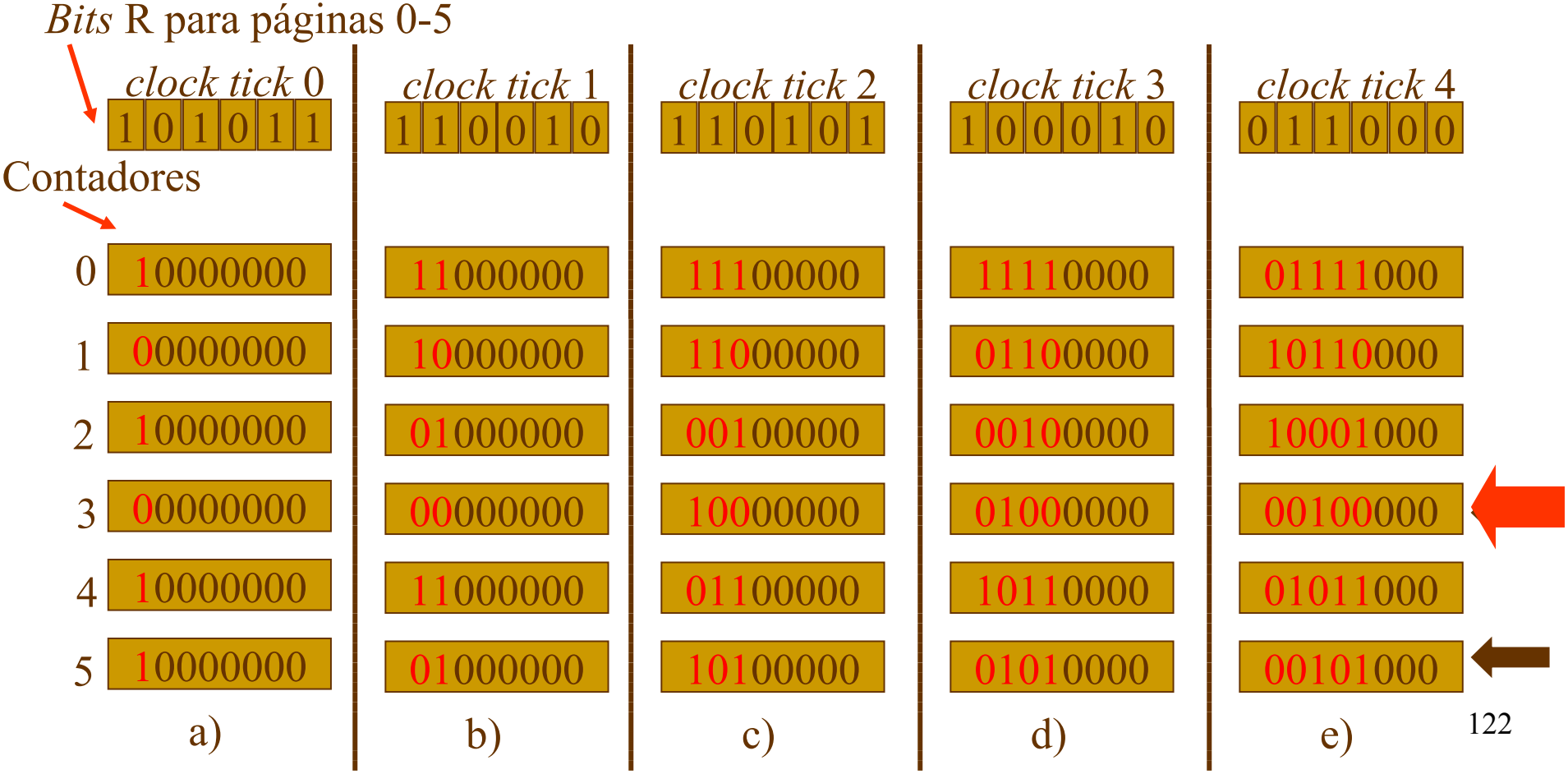
- Software: NFU
  - Para cada página existe um contador iniciado com zero e incrementado a cada referência à página;
  - Adiciona-se o valor do R (0 ou 1) ao contador
  - Página com menor valor do contador é candidata a troca;
  - O algoritmo jamais esquece nada
  - Problema: pode retirar páginas que já foram referenciadas e que poderão ser mais referenciadas no futuro
  - Compilador com vários passos:
    - Passo 1 pode levar a retirar aos dos passos seguintes cujos usos serão frequentes mais para frente

Gerenciamento de Memória
Troca de Páginas - Paginação

- Software: Algoritmo *aging*
  - Modificação do NFU, resolvendo o problema descrito anteriormente;
    - Além de saber **quantas vezes** a página foi referenciada, também controla **quando** ela foi referenciada;
    - Geralmente, 8 bits são suficientes para o controle se as interrupções de relógio (*clock ticks*) ocorrem a cada 20ms (10^{-3});

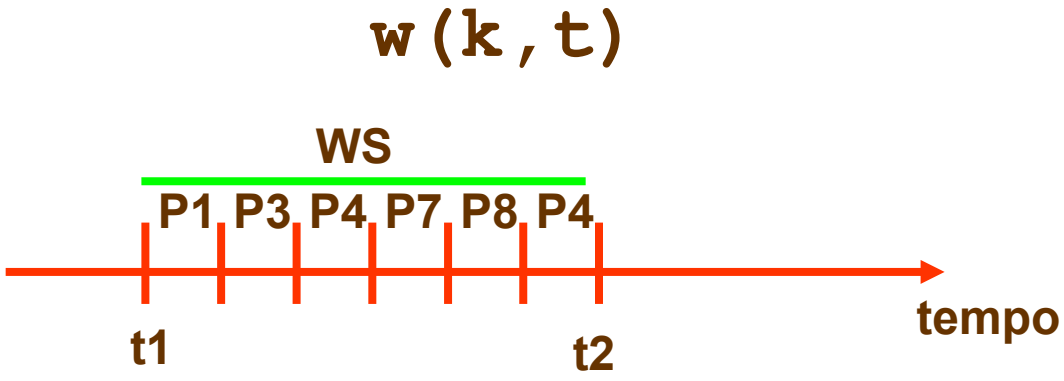
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *aging*



Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set (WS)*:
  - Paginação por demanda → páginas são carregadas na memória somente quando são necessárias;
  - Pré-paginação → *Working set*
    - Conjunto de páginas que um processo está efetivamente utilizando (referenciando) em um determinado tempo ***t***;



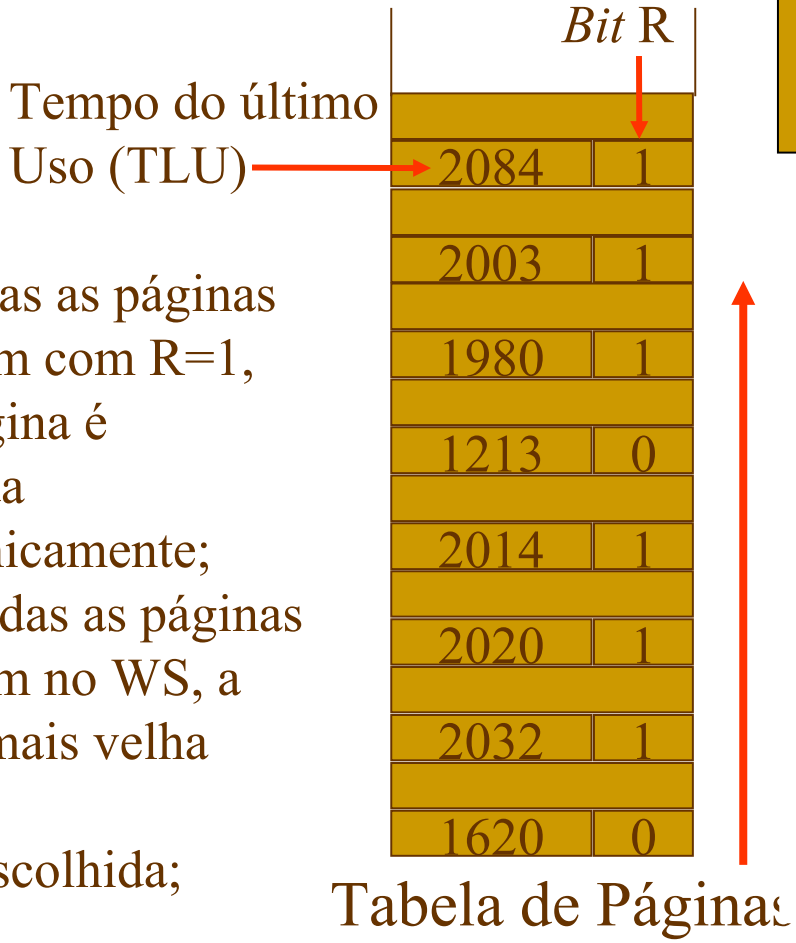
Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set (WS)*:
  - Objetivo principal: reduzir a falta de páginas
    - Um processo só é executado quando todas as páginas necessárias no tempo ***t*** estão carregadas na memória;
    - SO gerencia quais páginas estão no *Working Set;*
  - Para simplificar → o *working set* pode ser visto como o conjunto de páginas que o processo referenciou durante os últimos **t** segundos de tempo;
  - Utiliza *bit* R e o tempo de relógio (tempo virtual) da última vez que a página foi referenciada;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *Working Set*:

Tempo virtual atual (CVT): 2204
age = CVT – TLU
(Ex.: 2204-2084 = 120)
τ = múltiplos *clock ticks (window size)*



* Se todas as páginas estiverem com R=1, uma página é escolhida Randomicamente;
** Se todas as páginas estiverem no WS, a página mais velha com R=0 é escolhida;

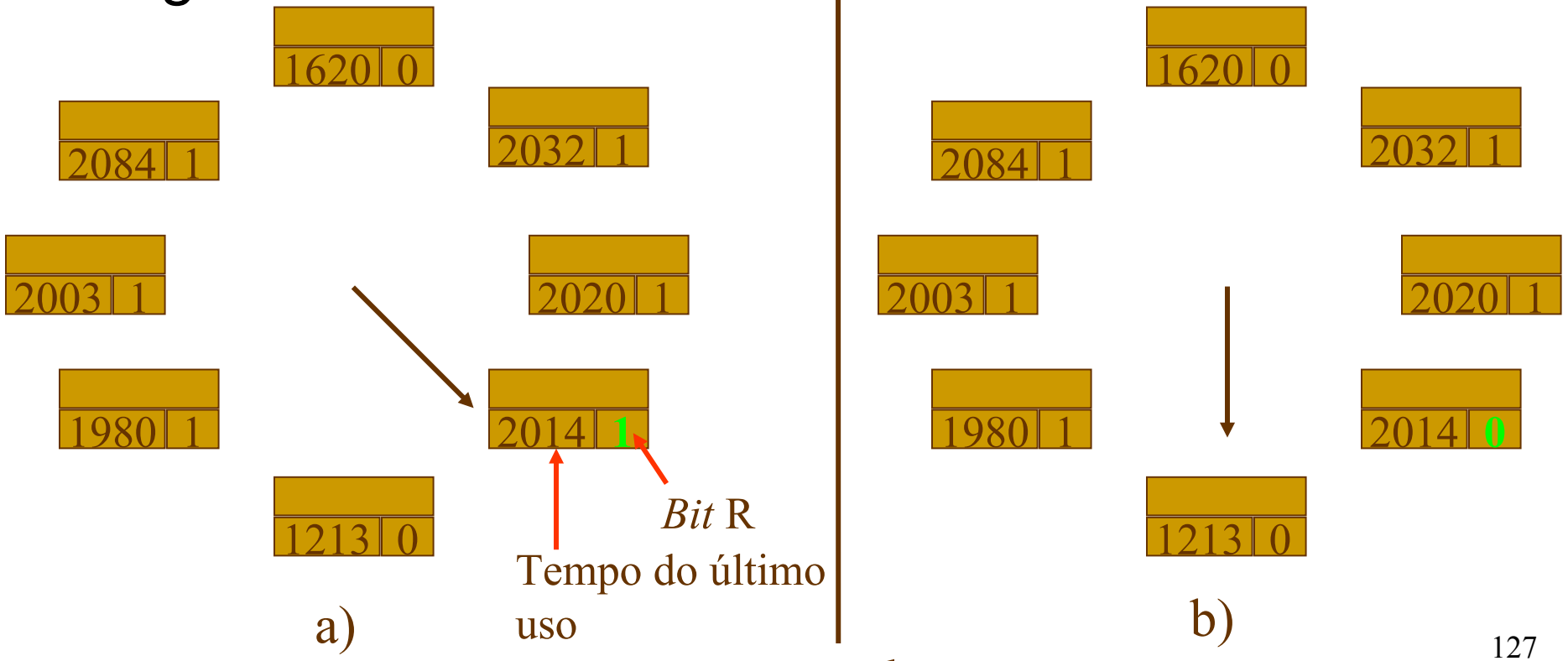
Percorrer as páginas examinando bit R;
Se (R==1)*
  página foi referenciada;
  faz TLU da página igual ao CVT;
Se (R==0 e *age* > τ)
  página não está no *working set*;
  remove a página;
Se (R==0 e *age* <= τ) **
  página está no *working set*;
  guarda página com maior *age*;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:
  - *Clock* + *Working Set*;
  - Lista circular de páginas formando um anel a cada página carregada na memória;
  - Utiliza *bit R* e o tempo da última vez que a página foi referenciada;
  - *Bit M* utilizado para agendar escrita em disco;

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:

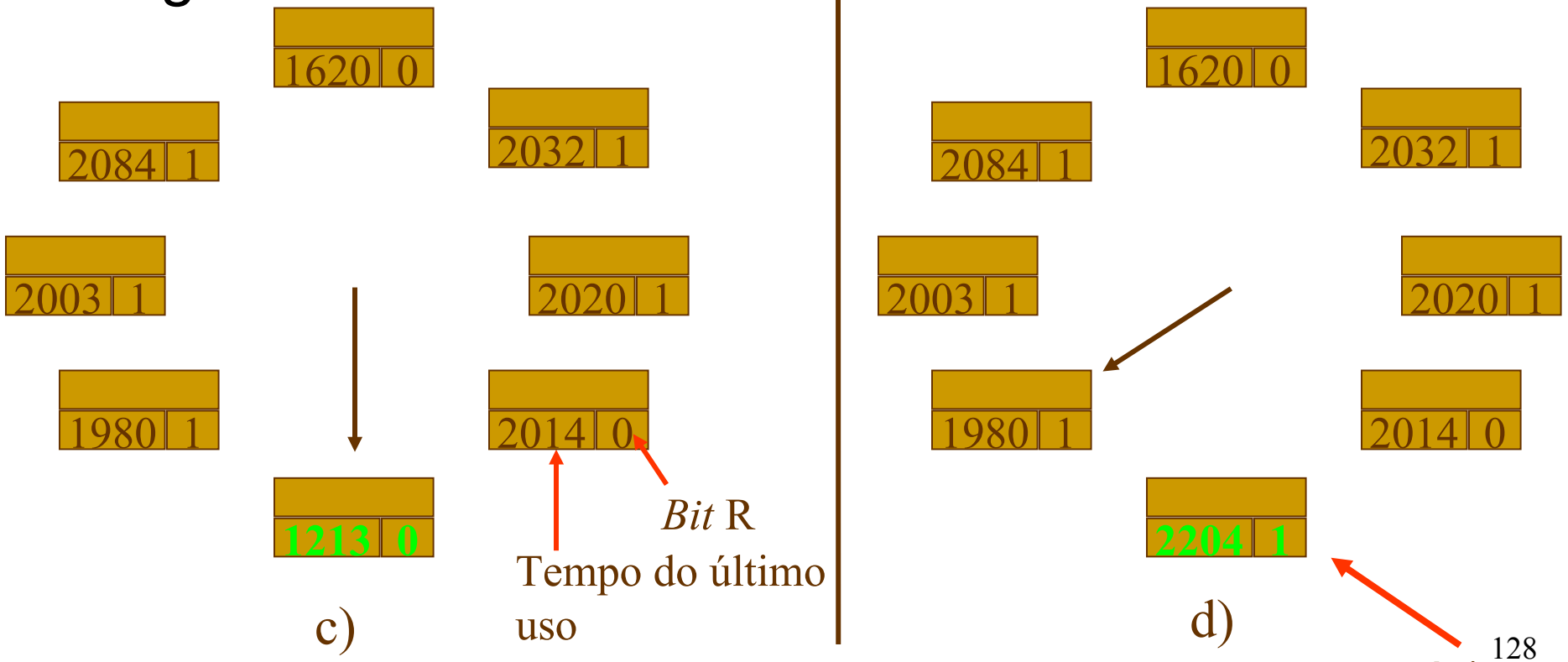


R==1

R=0 e ponteiro avança

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:

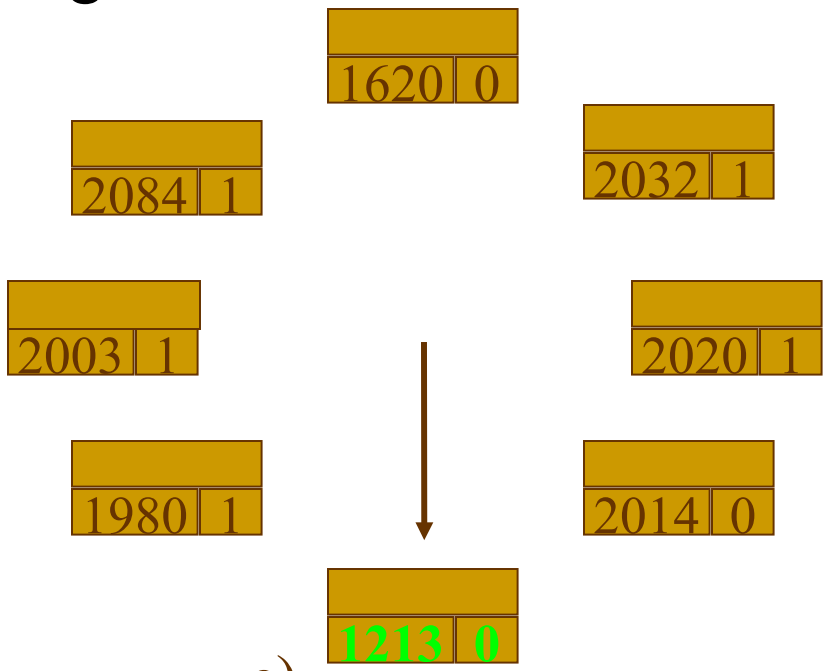


R==0 e age>t

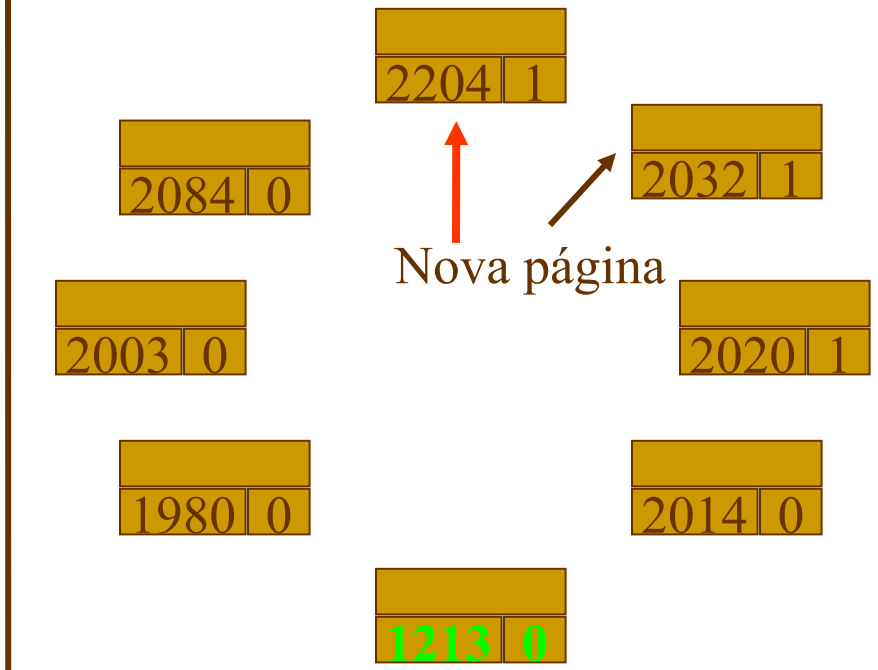
M==0 (não agenda escrita) → troca

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock*:



c)



d)

R==0 e age>t

M==1 (agenda escrita e continua procura)

Gerenciamento de Memória Troca de Páginas - Paginação

- Algoritmo *WSClock:*
  - Se todas estiverem com M==1; então escreve página atual no disco, e troca a página;
  - Melhor desempenho → menos acessos ao disco;

Gerenciamento de Memória
Troca de Páginas - Paginação

- Algoritmos de substituição local:
  - *Working Set*;
  - *WSClock*;
- Algoritmos de substituição local/global:
  - Ótimo;
  - NRU;
  - FIFO;
  - Segunda Chance;
  - LRU;
  - Relógio;

Gerenciamento de Memória
Implementação da Paginação

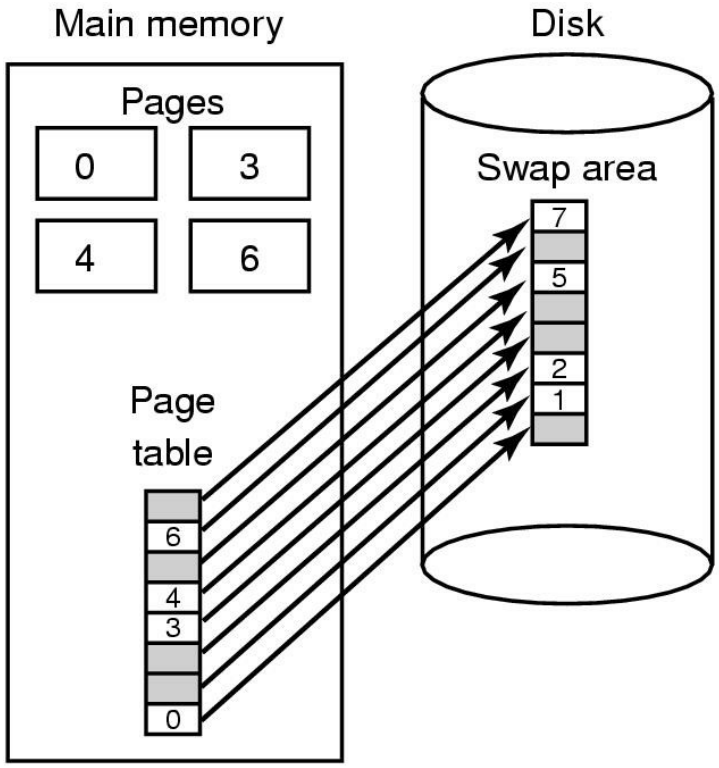
- **Memória Secundária – Disco**
  - A área de troca é gerenciada como uma lista de espaços disponíveis;
  - O endereço da área de troca de cada processo é mantido na tabela de processos;
    - Cálculo do endereço: MMU;
  - Possibilidade A - Assim que o processo é criado, ele é copiado todo para sua área de troca no disco, sendo carregado para memória quando necessário;
    - Área de troca diferente para dados, pilha e programa, pois a área de dados pode crescer e a área de pilha crescerá certamente;

Gerenciamento de Memória
Implementação da Paginação

- Memória Secundária – Disco
  - Possibilidade B - Nada é alocado antecipadamente, espaço é alocado em disco quando a página for enviada para lá.
  - Assim, processo na memória RAM não fica “amarrado” a uma área específica;

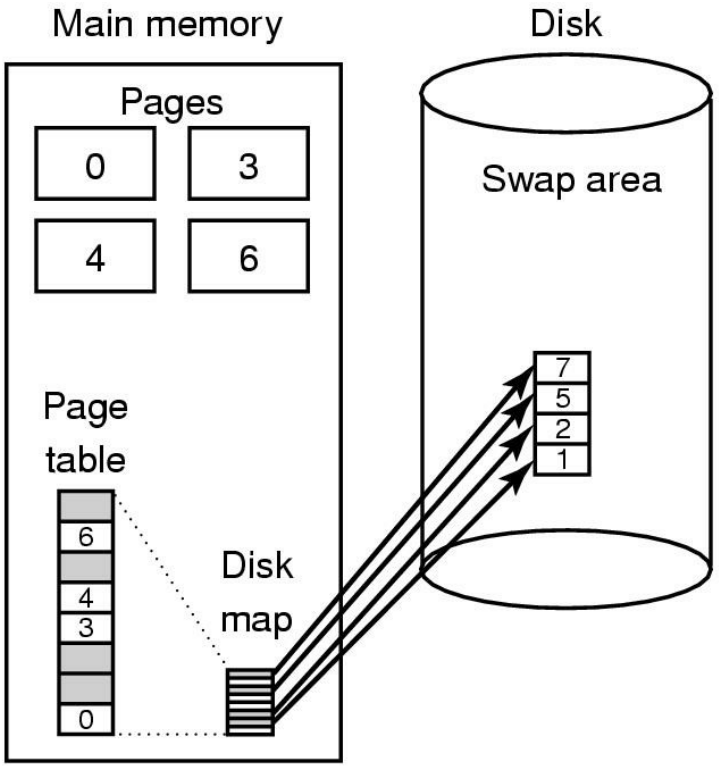
Gerenciamento de Memória
Implementação da Paginação

Como fica o disco – memória secundária



(a)

Área de troca estática



(b)

Área de troca dinâmica

Gerenciamento de Memória
Memória Virtual - Segmentação

- Segmentação: Visão do programador/compilador
  - Tabelas de segmentos com **n** linhas, cada qual apontando para um segmento de memória;
  - Vários espaços de endereçamento;
  - Endereço real → base + deslocamento;
  - Alocação de segmentos segue os algoritmos já estudados:
    - *FIRST-FIT*;
    - *BEST-FIT*;
    - *NEXT-FIT*;
    - *WORST-FIT*;
    - *QUICK- FIT*;

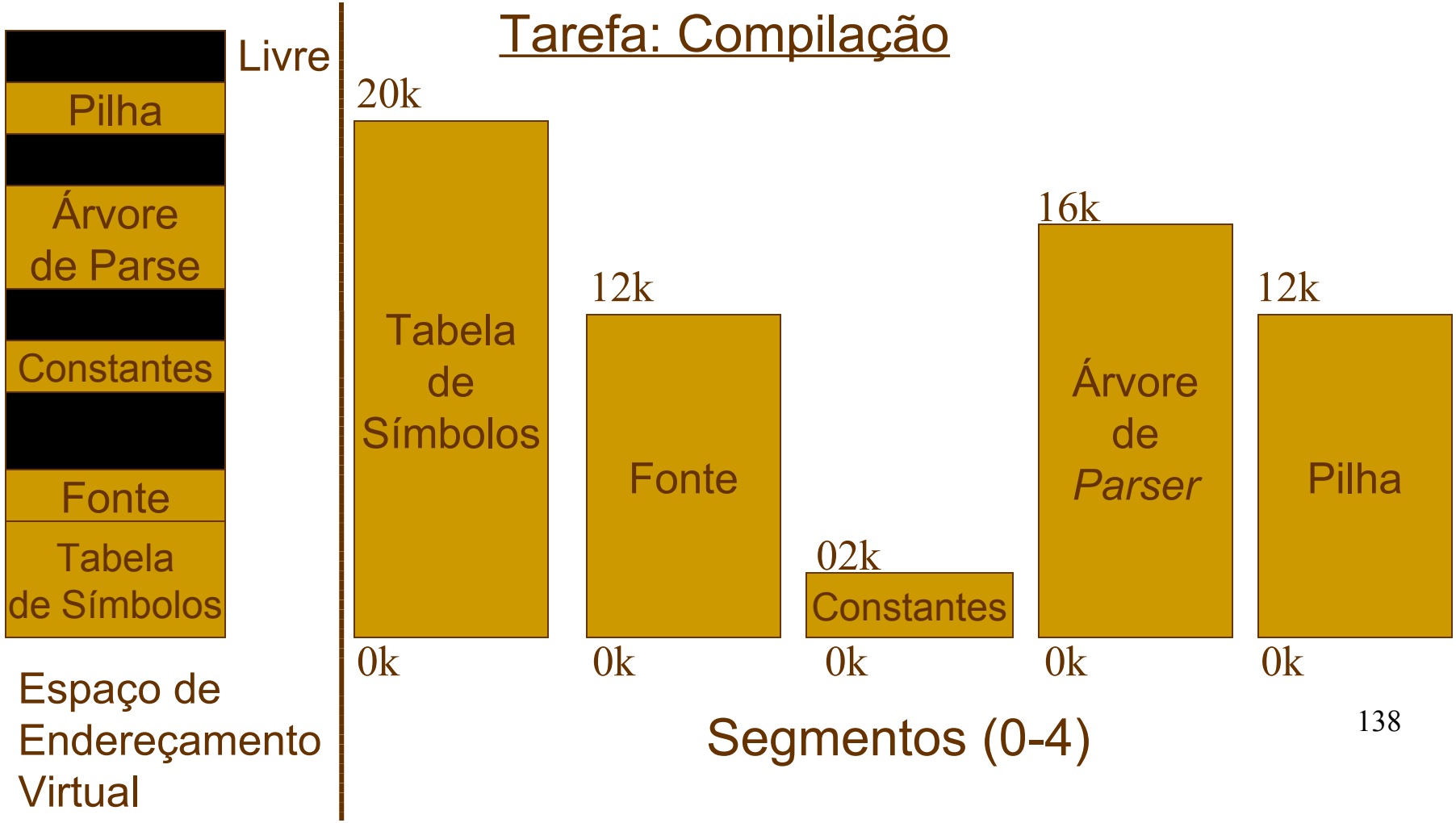
Gerenciamento de Memória
Memória Virtual - Segmentação

- Segmentação:
  - Facilita proteção dos dados;
  - Facilita compartilhamento de procedimentos e dados entre processos;
  - MMU também é utilizada para mapeamento entre os endereços lógicos e físicos;
    - Tabela de segmentos informa qual o endereço da memória física do segmento e seu tamanho;

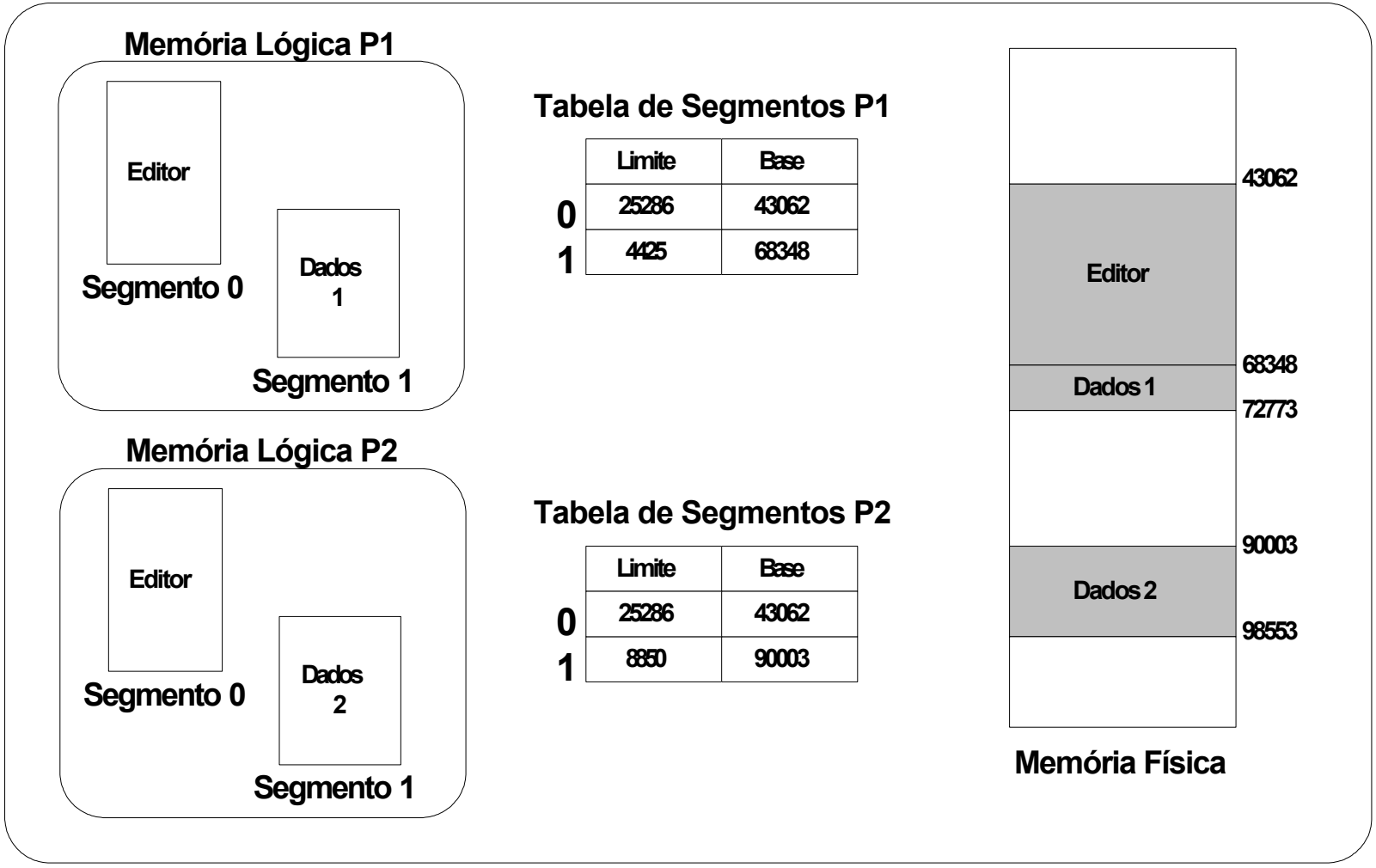
Gerenciamento de Memória
Memória Virtual - Segmentação

- Segmentação:
  - Problemas encontrados → embora haja espaço na memória, não há espaço contínuo:
    - Política de realocação: um ou mais segmentos são realocados para abrir espaço contínuo;
    - Política de compactação: todos os espaços são compactados;
    - Política de bloqueio: fila de espera;
    - Política de troca: substituição de segmentos;
  - Sem fragmentação interna, com fragmentação externa;

Gerenciamento de Memória
Memória Virtual - Segmentação

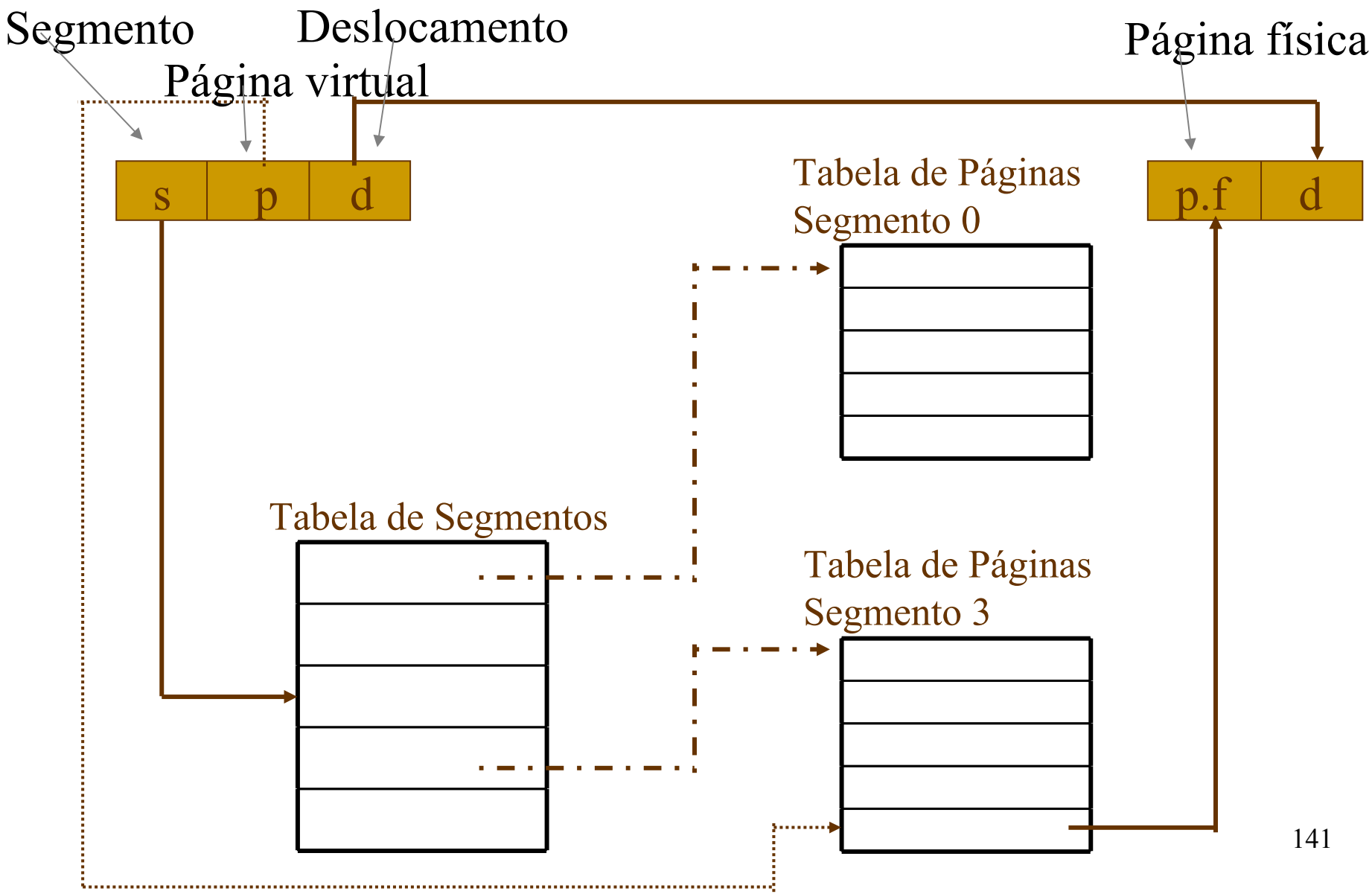


Gerenciamento de Memória Memória Virtual - Segmentação



Gerenciamento de Memória Segmentação-Paginada

- Espaço lógico é formado por segmentos
  - Cada segmento é dividido em páginas lógicas;
  - Cada segmento possui uma tabela de páginas →
    - mapear o endereço de página lógica do segmento em endereço de página física;
  - No endereçamento, a tabela de segmentos indica, para cada segmento, onde sua respectiva tabela de páginas está.


Segmento
Deslocamento
Página virtual
Página física
s
p
d
p.f
d
Tabela de Páginas
Segmento 0
Tabela de Segmentos
Tabela de Páginas
Segmento 3

Gerenciamento de Memória - *Thrashing*

- *Thrashing* (paginação excessiva)
  - Associado com o problema de definição do número de páginas/segmentos →

  troca de páginas/segmentos é uma tarefa cara e lenta;

  - Se o processo tiver um número de páginas muito reduzido, ele pode ficar muito tempo esperando pelo atendimento de uma falta de página →
    - muitos processos bloqueados;

Gerenciamento de Memória - *Thrashing*

- **Evitar o problema (paginação)**:
  - Taxa máxima aceitável de troca de páginas;
    - Suspender alguns processos, liberando páginas físicas (*swapping)*;
    - Risco de aumentar o tempo de resposta dos processos;
  - Determinar periodicamente o número de processos em execução e alocar para cada um, mesmo número de páginas;
    - Problema: processos grandes teriam o mesmo número de páginas de processos pequenos, causando paginação excessiva;

Gerenciamento de Memória - *Thrashing*

- Possível solução: Número de páginas proporcional ao tamanho do processo →
  - alocação dinâmica durante a execução dos processos
- PFF (*Page Fault Frequency*):
  - algoritmo informa quando aumentar ou diminuir a alocação de páginas de um processo
  - Controla também o tamanho do conjunto de alocação;

Gerenciamento de Memória
Memória Virtual

Consideração	Paginação	Segmentação
Programador deve saber da técnica?	Não	Sim
Espaços de endereçamento existentes	1	Vários
Espaço total de endereço pode exceder memória física?	Sim	Sim
É possível distinguir procedimento de dados e protegê-los?	Não	Sim

Gerenciamento de Memória
Memória Virtual

Consideração	Paginação	Segmentação
Tabelas de tamanho variável podem ser acomodadas sem problemas?	Não	Sim
Compartilhamento de procedimentos entre usuário é facilitado?	Não	Sim
Por que?	Para obter espaço de endereçamento maior sem aumentar memória física	Para permitir que programas e dados possam ser divididos em espaços de endereçamento logicamente independentes; compartilhamento e proteção

Perguntas?

- Ler o capítulo sobre a gerência de memória