Capítulo

11

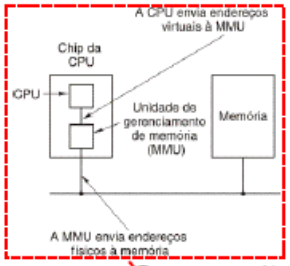
Gerenciamento de memória virtual

Sumário

PEARSON Prentice Hall

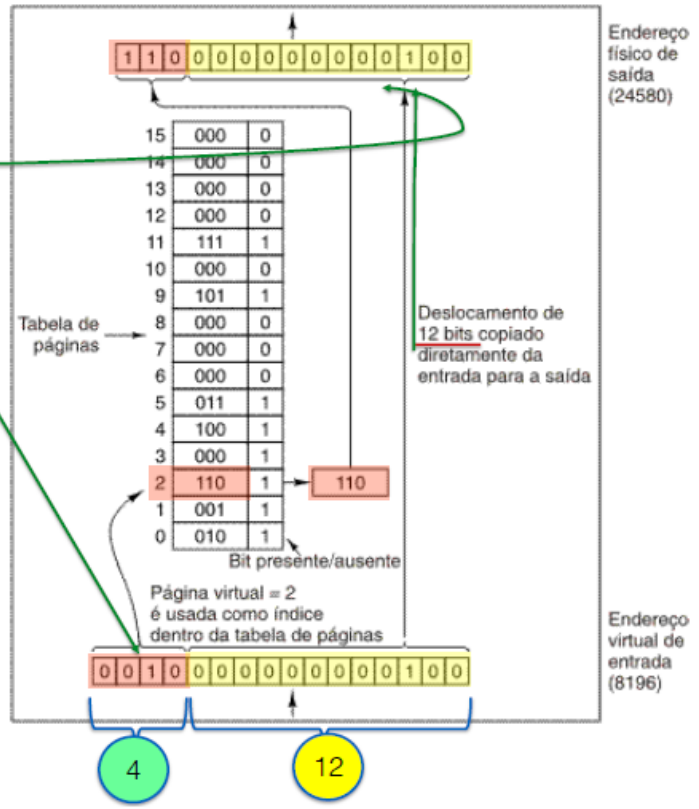
Objetivos

- **Este capítulo apresenta:**
  - Introdução
  - Calculando o tamanho de uma tabela de página
  - Tabelas de páginas Multinível, Hash e Invertidas
  - Alocações de páginas: Fixa e Variável
  - Busca de páginas
  - Substituição aleatória de páginas: Local e Global
  - Comparação entre paginação e segmentação



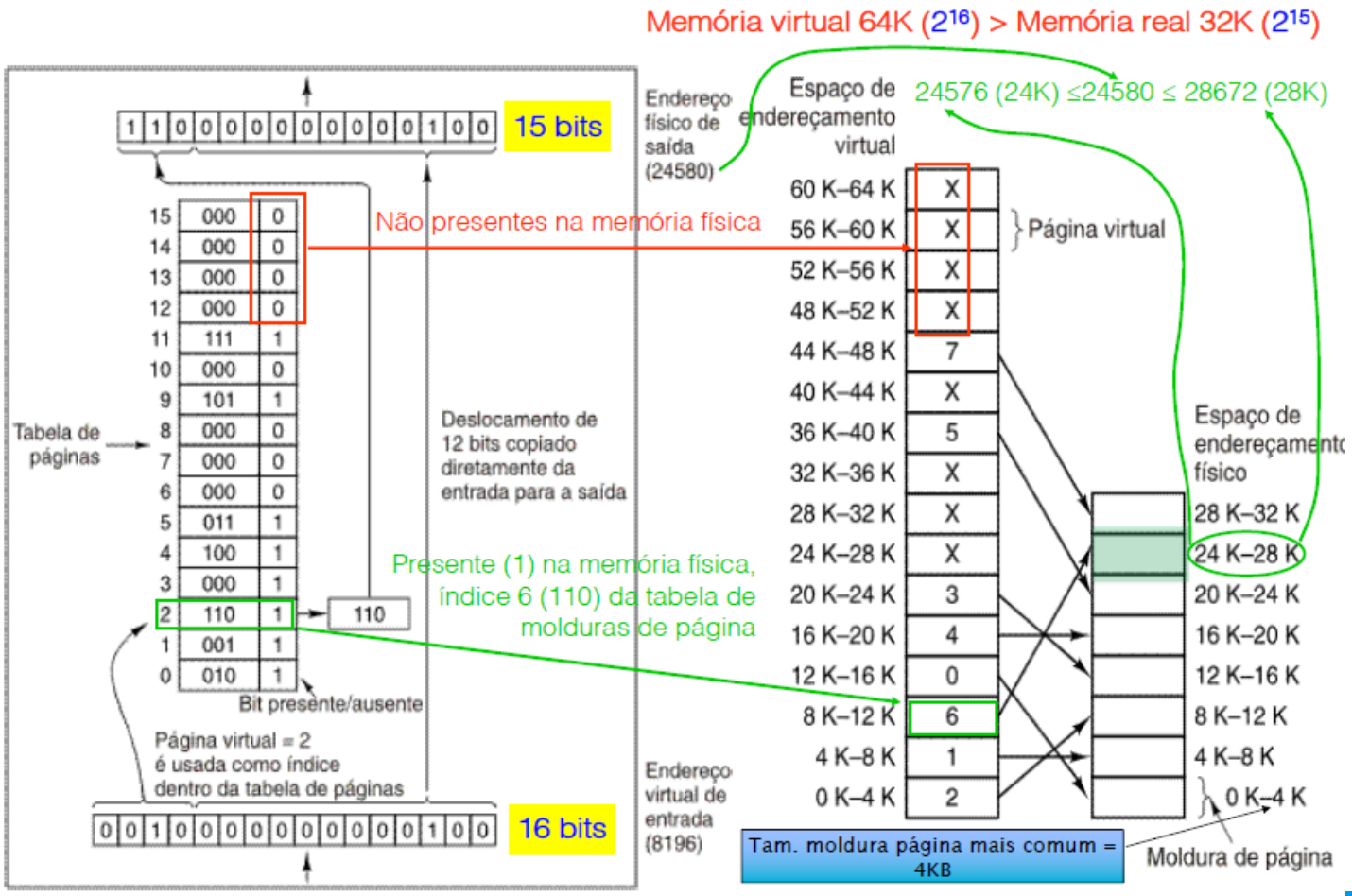
Paginação

- Operação interna da MMU com 16 (2^4) páginas de 4KB (4096 = 2^{12}): determinação do endereço físico de memória em função do endereço virtual vindo da CPU (PC) e da tabela de páginas
- Tabela de páginas fica dentro da MMU



11.1 Introdução

Paginação: Relação entre endereços virtuais e endereços de memória física dados pela tabela de páginas



11.1 Introdução

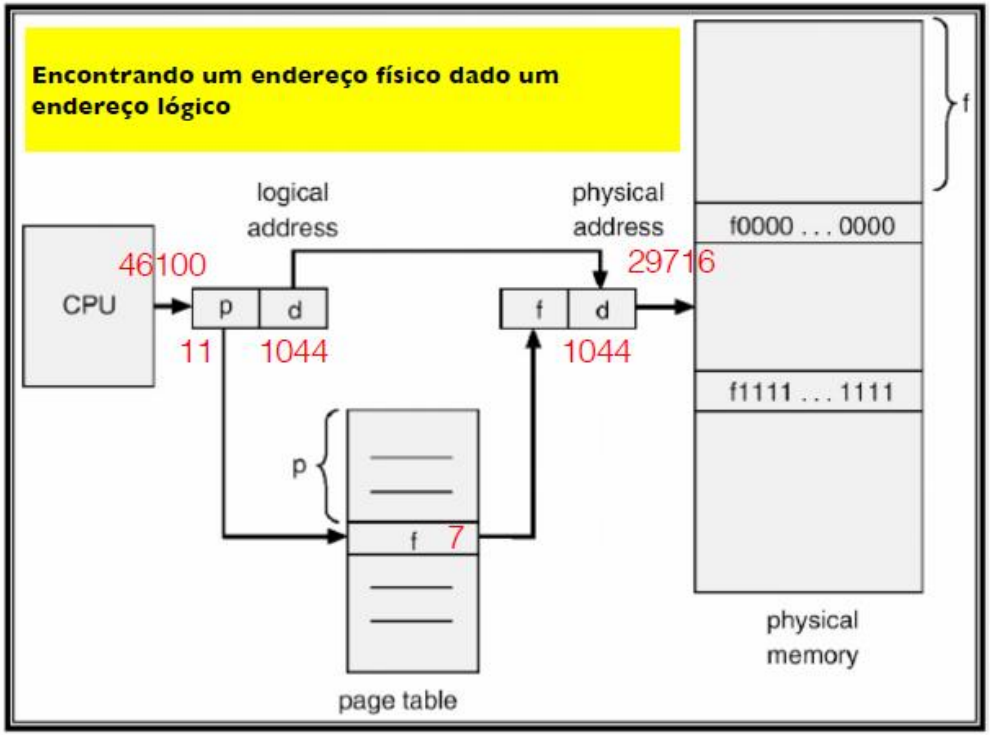
Assumindo tamanho (da moldura) de página = 4KB (4096 B)

Ex.: endereço de referência (lógico) = 46100

p = 46100 div 4096 = 11; d = 46100 mod 4096 = 1044,
onde p é a entrada na tabela de página e d é o deslocamento (*displacement*)

Endereço físico = f * tamanho de página + d, onde f é o número da moldura de página física

Endereço físico = 7 * 4096 + 1044 = 29716 ⇨ Endereço na moldura 7 da memória (física)



11.2 Calculando o tamanho de uma tabela de página

- **Considerando um sistema com:**
  1. logical address space: 32-bit
  2. page size: 4KB (2^{12})
  3. page table entry size: 4 bytes (32 bits)
  4. physical memory: 2GB

Tamanho de uma tabela de páginas = no._entradas * tamanho_entrada

PEARSON Prentice Hall

11.2 Calculando o tamanho de uma tabela de página

- Um endereço lógico de 32 bits, considerando um tamanho de páginas de 4KB (2^{12}), é particionado como:
  - <$2^{32}/2^{12}$ bits por no. página, 12 bits de deslocamento dentro da página> = <20 bits por no. página, 12 bits de deslocamento>
- Assim, o número de entradas na tabela de páginas (total de números de páginas virtuais) = 2^{20}
- Ou seja, o **tamanho de uma tabela de páginas = no._entradas * tamanho_entrada** = 2^{20} * 4 bytes (32 bits) = 4.194.304 = 4MB (por processo)

No sistema considerado, suponha **100 processos**, por exemplo:
se cada processo possui uma tabela de páginas de 4MB, apenas as tabelas ocuparão **400MB** de um total de 2GB, ou seja, cerca de 20% do espaço da memória física...

11.3 Acelerando a Paginação

1. O mapeamento de endereço virtual para endereço físico deve ser rápido
2. Se o espaço de endereçamento virtual for grande, a **tabela de páginas será grande...**

Uma TLB para acelerar a paginação.

Válida	Página virtual	Modificada	Proteção	Quadro de página
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
1	860	1	RW	14
1	861	1	RW	75

11.4 Tabela de páginas para memória grande

As TLBs podem ser usadas para acelerar a tradução de endereços virtuais para endereços físicos em relação ao esquema de tabela de páginas na memória original.

1. Mas esse não é o único problema que precisamos combater.
2. Outro problema é como lidar com espaços de endereços virtuais muito grandes.

Como a tabela de páginas é organizada?

- Tabelas de páginas multinível ou Paginação hierárquica
- Tabelas de página com hash
- Tabelas de página invertidas

PEARSON Prentice Hall

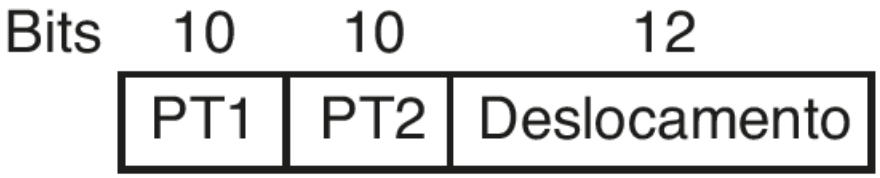
11.4.1 Tabelas de páginas multinível ou hierárquica

- Quebre o espaço de endereço lógicos em múltiplas tabelas de páginas;
- Uma técnica simples é uma tabela de páginas em dois níveis;
- **Manter apenas a parte da tabela necessária**
- Ocupa menos espaço para o próprio S.O.

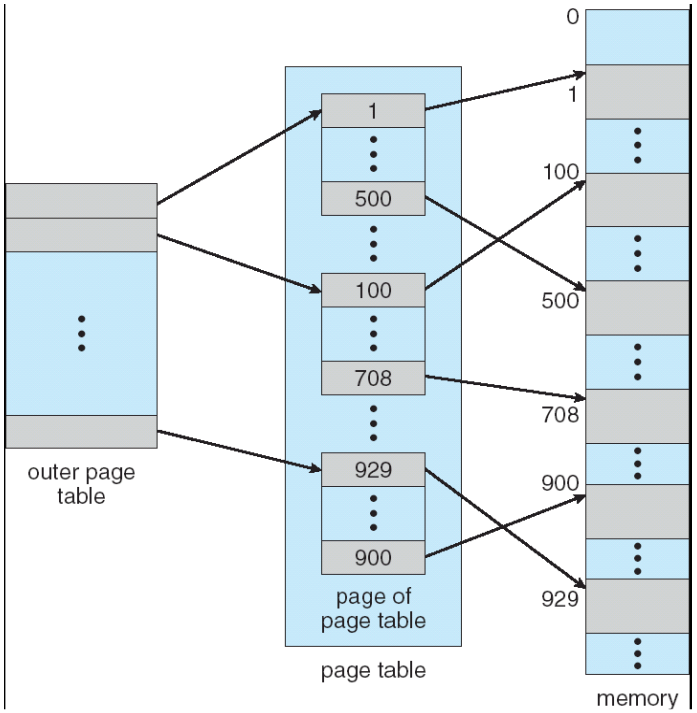
PEARSON Prentice Hall

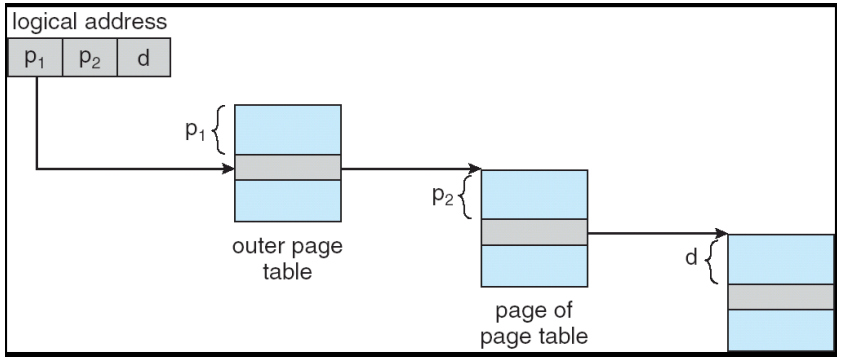
11.4.1 Tabelas de páginas multinível ou hierárquica

- **Um endereço lógico (em máquinas de 32 bits com tamanho de página de 4K) é dividido em:**
  - **um número de página** contendo 20 bits
  - **um deslocamento** de página contendo 12 bits
- **Como a tabela de página é paginada, o número de página é dividido ainda em:**
  - Número de página PT1 (10 bits) – Índice da **tabela mais externa**.
  - Número de página PT2 (10 bits) – **Deslocamento da tabela mais externa**.
- **Assim, um endereço lógico é o seguinte:**



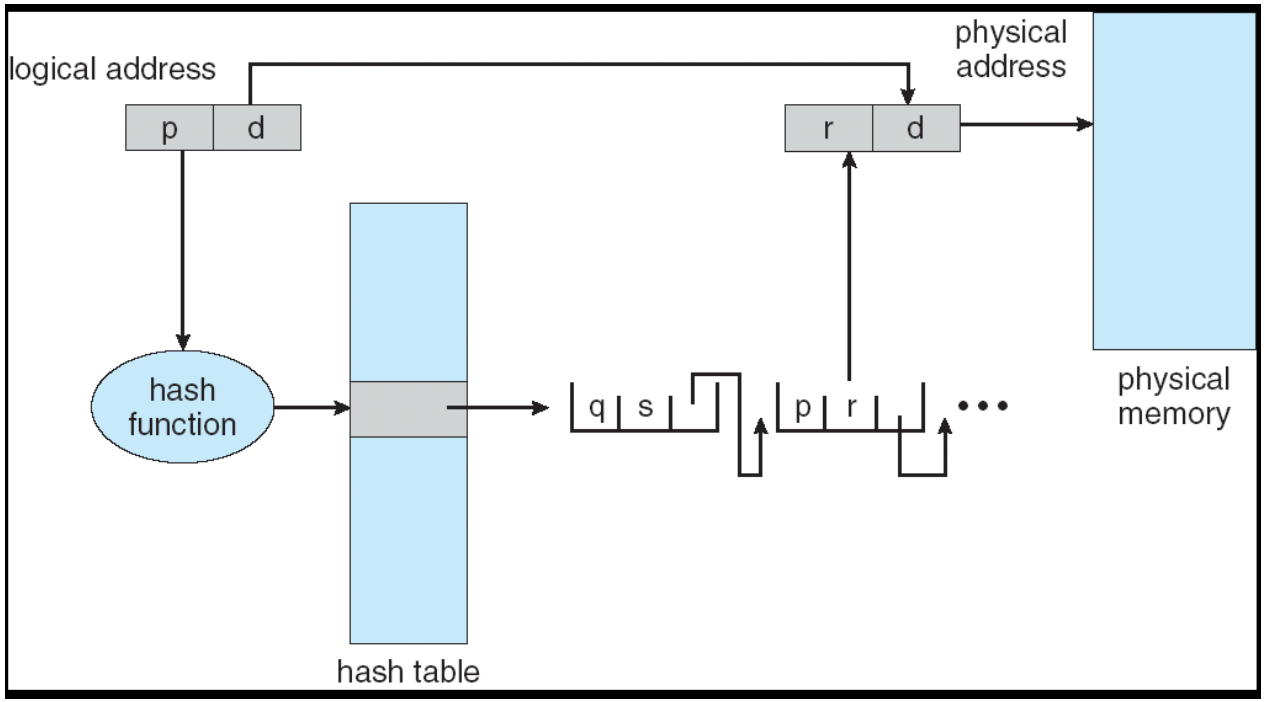
11.4.1 Tabelas de páginas multinível ou hierárquica





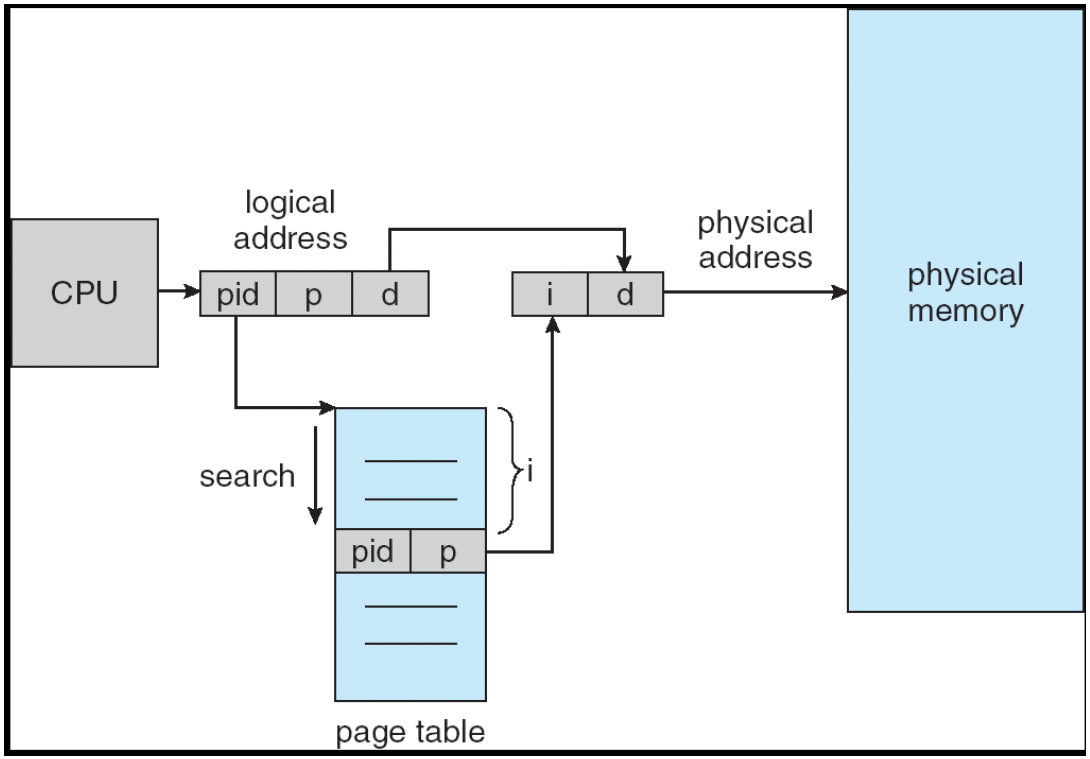
11.4.2 Tabelas de página com Hash

- O número de página virtual é usado para função hash.
- Cada entrada na tabela de página contém uma lista ligada de elementos, que consiste em:
  - O # da página virtual;
  - O # da moldura (frame).
  - Um ponteiro para o próximo.



11.4.3 Tabelas de páginas Invertida

- Possui uma entrada por moldura de memória física.
  - **Em vez de uma entrada por página no espaço virtual;**
  - **A entrada inclui o processo e a página virtual;**
  - **O deslocamento na leitura é o índice do frame;**
- Poupam muito espaço quando o espaço virtual é muito maior que o físico.



11.4.3 Tabelas 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;

PEARSON Prentice Hall

11.5 Alocação de páginas: 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:**
  - (i) 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.

PEARSON Prentice Hall

11.6 Alocação de páginas: Variável ou Dinâmica

- Número máximo de páginas reais alocadas ao processo varia durante sua execução;
- **Vantagens:**
  - (i) processos com elevada taxa de paginação podem ter seu limite de páginas reais ampliado;
  - (ii) processos com baixa taxa de paginação podem ter seu limite de páginas reais reduzido.
- **Desvantagem:**
  - (i) monitoramento constante.

11.7 Busca de páginas

- **Paginação simples**
  - Todas as páginas virtuais do processo são carregadas para a memória principal;
  - Assim, sempre todas as páginas são válidas.

- **Paginação por demanda**
  - Processos começam com nenhuma página na memória.
  - Assim que a CPU tenta executar a primeira instrução gera um page fault;
  - O Sistema Operacional traz a página que falta à memória.

Substituição aleatória de páginas

Estratégia de substituição de páginas de baixa sobrecarga que não diferencia um determinado processo de outro.

Toda página na memória principal tem a mesma probabilidade de ser selecionada para substituição.

A página a ser referenciada em seguida poderia facilmente ser selecionada como a página seguinte a ser substituída.

11.8.1 Substituição de página local *versus* global

- **Implementação de um sistema de memória virtual paginado**

  Estratégias de substituição de página global: aplicadas a todos os processos como uma unidade.

  Tendem a ignorar as características do comportamento individual de um processo.

  Estratégia de substituição de páginas MRU global (gMRU):

  Substitui a página menos recentemente usada em todo o sistema.

  Estratégia global de substituição de páginas SEQ (sequência):

  Usa a estratégia MRU para substituir páginas até que a sequência de falta de página em páginas contíguas seja detectada, ponto em que utiliza a estratégia de substituição de página 'usada mais recentemente' (UMR).

  Estratégia de substituição de páginas local: considera cada processo individualmente.

  Permite que o sistema ajuste a alocação de memória de acordo com a importância relativa de cada processo a fim de melhorar o desempenho.

11.8.1 Substituição de página local *versus* global

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

A0
A1
A2
A3
A4
A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

(b)

A0
A1
A2
A3
A4
A5
B0
B1
B2
A6
B4
B5
B6
C1
C2
C3

(c)

11.9 Comparação entre paginação e segmentação

Consideração	Paginação	Segmentação
O programador precisa saber que essa técnica está sendo usada?	Não	Sim
Há quantos espaços de endereçamento linear?	1	Muitos
O espaço de endereçamento total pode superar o tamanho da memória física?	Sim	Sim
Rotinas e dados podem ser distinguidos e protegidos separadamente?	Não	Sim
As tabelas cujo tamanho flutua podem ser facilmente acomodadas?	Não	Sim
O compartilhamento de rotinas entre os usuários é facilitado?	Não	Sim
Por que essa técnica foi inventada?	Para obter um grande espaço de endereçamento linear sem a necessidade de comprar mais memória física	Para permitir que programas e dados sejam divididos em espaços de endereçamento logicamente independentes e para auxiliar o compartilhamento e a proteção

10.7 Algoritmos de substituição de páginas

Os algoritmos de substituição de página ocorre como estratégia de realocação.

Algoritmo	Comentário
Ótimo	Não implementável, mas útil como um padrão de desempenho
NRU (não usado recentemente)	Aproximação muito rudimentar do LRU
FIFO (primeiro a entrar, primeiro a sair)	Pode descartar páginas importantes
Segunda chance	Algoritmo FIFO bastante melhorado
Relógio	Realista
LRU (usada menos recentemente)	Excelente algoritmo, porém difícil de ser implementado de maneira exata
NFU (não frequentemente usado)	Aproximação bastante rudimentar do LRU
Envelhecimento (*aging*)	Algoritmo eficiente que aproxima bem o LRU
Conjunto de trabalho	Implementação um tanto cara
WSClock	Algoritmo bom e eficiente

PEARSON Prentice Hall