Instituto de Informática - UFRGS

Sistemas Operacionais

Paginação

Aula 13

Introdução

- Problema com alocação particionada
  - Necessidade de uma área contígua de memória (tamanho do processo)
- Solução:
  - Retirar a restrição de que o espaço de endereçamento deva ser contíguo
- "Desacoplar" a ideia do espaço de endereçamento lógico do espaço de endereçamento físico
  - Necessidade de "mapear" o espaço lógico no espaço físico
  - Dois métodos básicos:
    - Paginação
    - Segmentação

Instituto de Informática - UFRGS
A. Carissimi -23-avr.-12

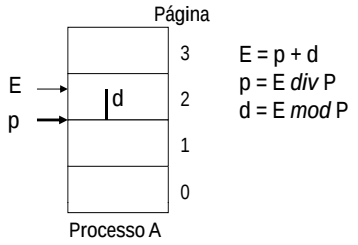
Princípio básico da paginação

- Divisão do espaço de endereçamento em blocos de tamanho fixo
  - São os quadros (*frames*) para o espaço de endereçamento físico (RAM)
  - São as páginas para o espaço de endereçamento lógico (processo)
- Regra:
  - Uma página pode ser carregada em qualquer quadro livre
    - "Quebra" a noção de espaço contíguo
    - Necessário mapear endereço lógico em físico
- Importante:
  - Processo "enxerga" um espaço de endereçamento contíguo, embora ele seja não contíguo
    - Noção de endereço lógico versus endereço fisico

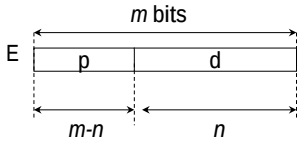
Instituto de Informática - UFRGS
A. Carissimi -23-avr.-12

Espaço de endereçamento lógico

- Endereço lógico é dividido em duas componentes:
  - Número da página
  - Deslocamento dentro de uma página
- Páginas podem assumir qualquer tamanho, porém emprega-se um tamanho potência de 2 para facilitar operações *div* e *mod*



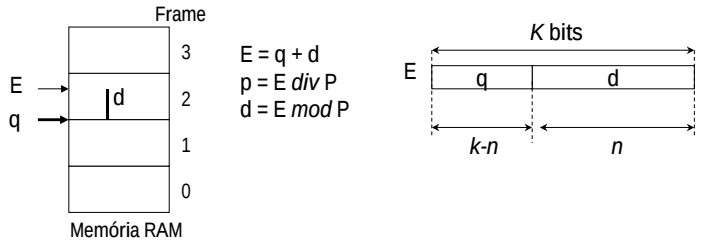
E = p + d
p = E *div* P
d = E *mod* P



Instituto de Informática - UFRGS
A. Carissimi -23-avr.-12

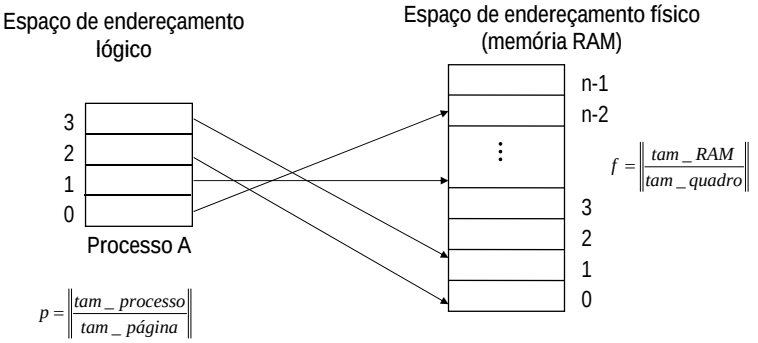
Espaço de endereçamento físico

- Endereço físico é dividido em duas componentes:
  - Número do quadro
  - Deslocamento dentro do quadro
- Quadros possuem o mesmo tamanho da página

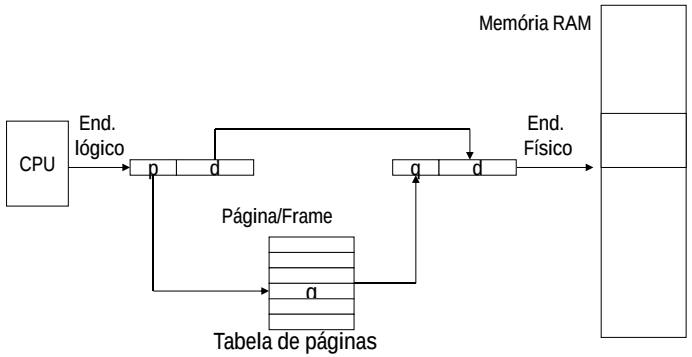


Mapeamento de páginas em quadros

- Como uma página pode ser carregada em qualquer quadros é necessário mapear páginas em quadros
  - Tabela de páginas

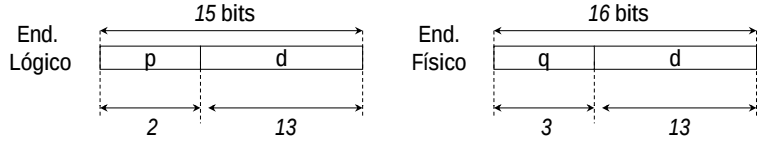


Tradução de endereço lógico em endereço físico

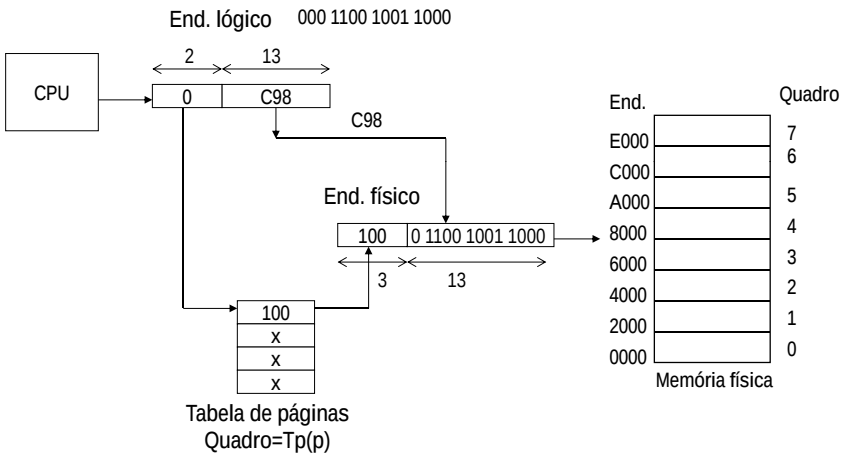


Exemplo de paginação

- Características do sistema:
  - Memória física: 64 kbytes (16 bits)
  - Tamanho processo (máx): 32 kbytes (15 bits)
  - Páginas 8 kbytes
- Paginação:
  - Número de quadros: 64/8 = 8 (0 a 7) → 3 bits
  - Número de páginas: 32/8 = 4 (0 a 3) → 2 bits
  - Deslocamento: 8 kbytes → 13 bits

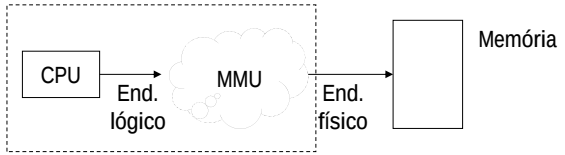


Exemplo de paginação (*cont.*): tradução



Alguns detalhes...

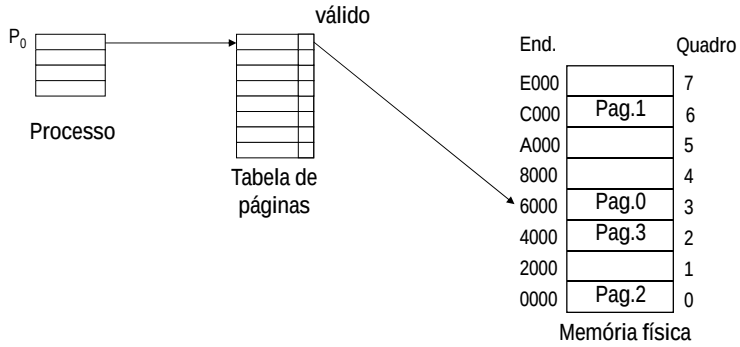
- A paginação elimina a fragmentação externa
  - Apresenta fragmentação interna (limitada a última página)
    - Fragmentação interna média = (tam_página – 1)/2
    - Observação: pode ser por área (código, dados, heap e pilha)
- Mapeamento é feito com auxílio do hardware do processador
  - Memory Management Unit



Proteção

- Proteção de acesso é garantida por definição:
  - Processos acessam somente suas páginas → end. válidos
  - Endereço inválido apenas na última página
    - Se houver fragmentação interna
- Inclusão de bits de controle na tabela de página (por entrada)
  - Bit de validade:
    - Página pertence ou não ao end. lógico do processo
  - Controle de acesso
    - Indicação se a página é de leitura, escrita ou executável
    - Páginas apenas de leitura, página de leitura/escrita

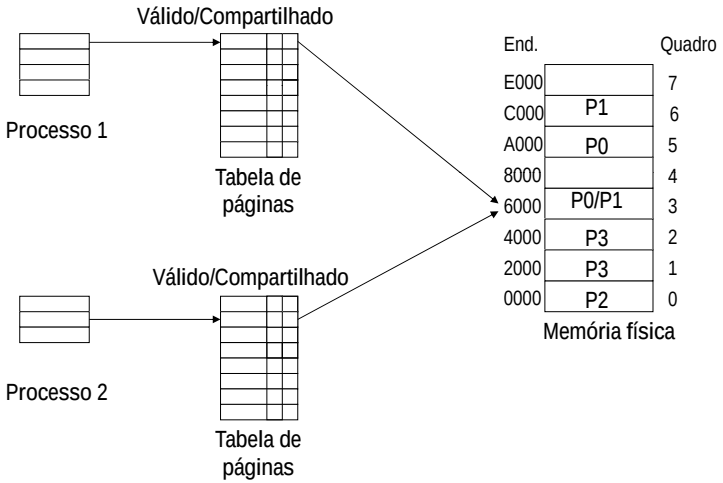
Exemplo de proteção



Compartilhamento de páginas

- Código compartilhado
  - Uma cópia do código (*read-only*, reentrante) pode ser compartilhada entre vários processos (e.g.; editores de texto, compiladores, etc...)
  - O código compartilhado pertence ao espaço lógico de todos os processos
- Dados e código próprios
  - Cada processo possui sua própria área de código e seus dados

Exemplo de compartilhamento



Problema com a tabela de páginas

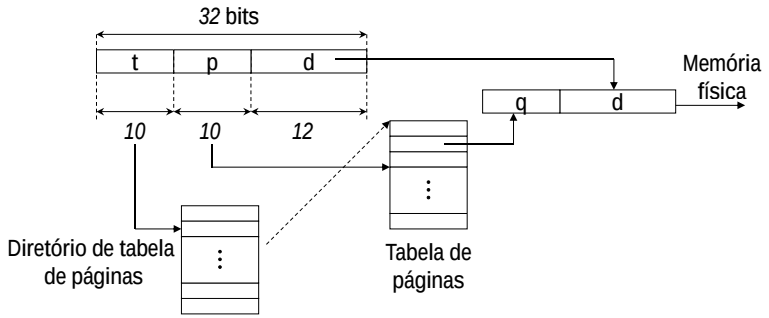
- Como dimensionar o tamanho da tabela de páginas?
  - Tabelas de páginas tendem a ser grandes
  - Agravado em novos processadores (e.g. processador de 64 bits, com páginas de 4 KB, gera uma tabela 2^{52} entradas)
- Tamanho fixo:
  - Limita o tamanho máximo de um processo
  - Desperdício de memória: aloca *n* entradas para usar *k* entradas (k << n)
- Tamanho variável:
  - Dimensiona conforme necessidade, mas como armazenar a tabela de páginas?
    - Contíguo em memória → fragmentação externa
    - Paginando a própria tabela
  - Estratégia normalmente usada pelos sistemas operacionais modernos

Paginação multinível

- Solução para o dimensionamento da tabela de páginas
- Organiza a tabela de páginas em dois níveis hierárquicos
  - Diretório de tabela de páginas
  - Tabela de páginas propriamente dita
- Diretórios de tabela de páginas podem ser em *n* níveis
  - Uma entrada do diretório da tabela de páginas de nível *n* aponta para um diretório de tabelas de páginas de nível *n-1*
  - O último nível é a tabela de páginas

Exemplo: Paginação em dois níveis

- Processadores 80x86
  - End. Lógico: 4 Gbytes (32 bits)
  - Páginas: 4 kbytes
  - Tamanho da tabela de páginas: 4 Gbytes / 4 kbytes = 1048576 entradas



Paginação em três níveis

- Típico de arquiteturas de processadores de 64 bits

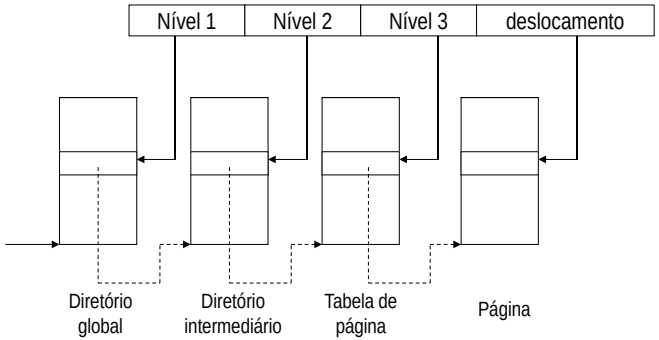
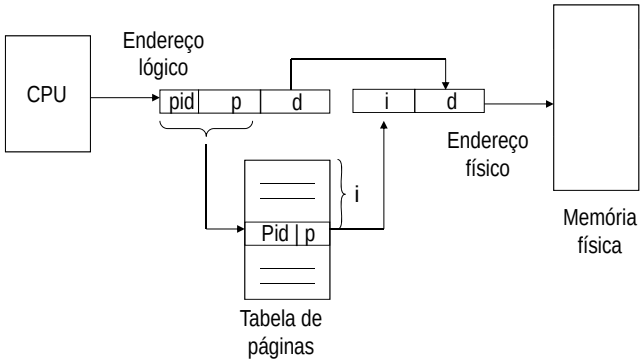


Tabela de páginas invertida

- Outra abordagem para o problema do tamanho da tabela de páginas
- Tabela de páginas invertida:
  - Uma tabela de páginas para todo o sistema (não mais por processo)
  - Uma entrada para cada quadro
  - Endereço lógico da página e a qual processo pertence
- Endereço lógico é formado por <*process_id*, página, deslocamento>
- Cada entrada da tabela possui < *process_id*, página>
- Tabela é pesquisada e retorna, se presente, o índice *i* associado a entrada
  - Cada indice corresponde a um quadro

Esquema da tabela de páginas invertida



Características da paginação (resumo)

- Paginação é um tipo de relocação (via hardware)
- Não apresenta fragmentação externa, porém gera fragmentação interna (restrita apenas a última página)
- Importante:
  - Visão do usuário: espaço de endereçamento contíguo
  - Visão do sistema: processo é "esparramado" na memória física
- *n* páginas são alocadas a *n* quadros implicando na criação de uma função de mapeamento
  - O maior processo e a quantidade de processos em memória (escalonador de curto prazo) é limitado pela capacidade da RAM (quadros)
    - Solução para isso é empregar memória virtual (será visto mais adiante)
- Facilita implementação de proteção e compartilhamento

Leituras complementares

- A. Tanenbaum. *Sistemas Operacionais Modernos* (3ª edição), Pearson Brasil, 2010.
  - Capítulo 3: seções 3.3.1 e 3.3.2
- A. Silberchatz, P. Galvin; *Sistemas Operacionais*. (7ª edição). Campus, 2008.
  - Capítulo 8 (seções 8.4 e 8.5)
- R. Oliveira, A. Carissimi, S. Toscani; *Sistemas Operacionais*. Editora Bookman 4ª edição, 2010
  - Capítulo 6 (seção 6.5)