Sistemas Operacionais

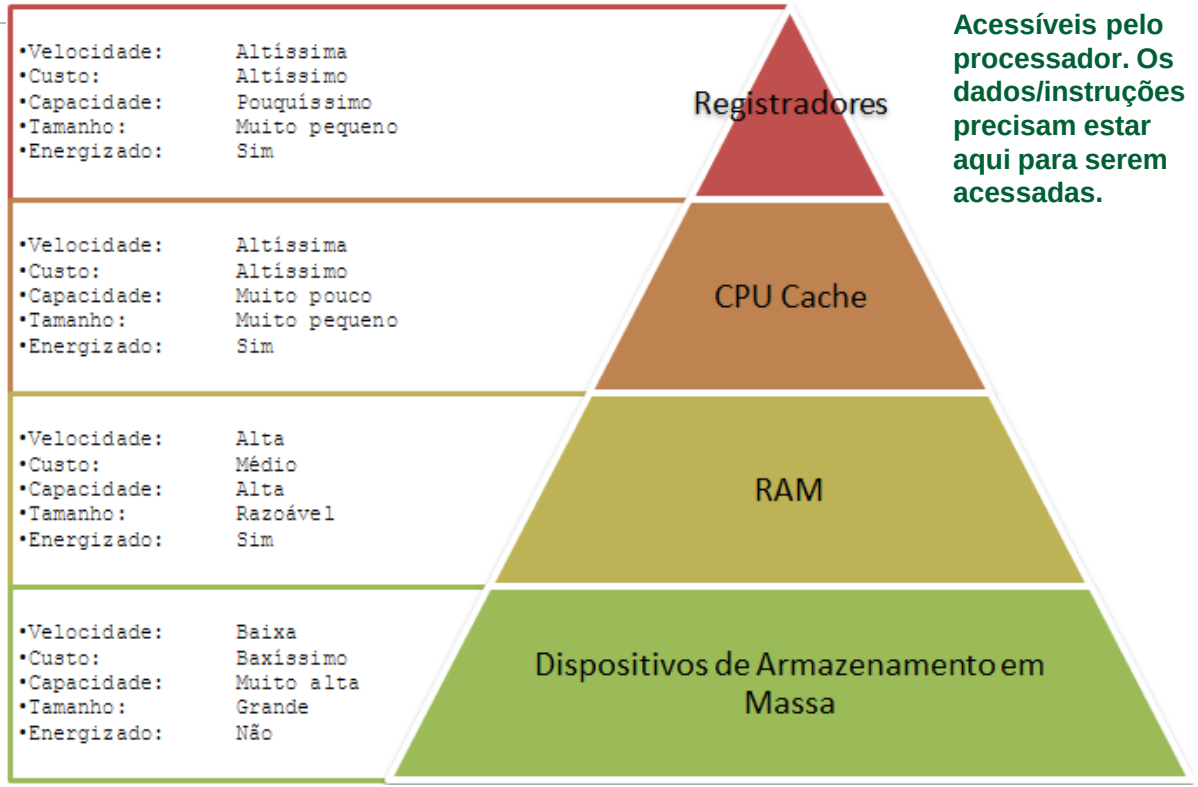
Aula 16 - Memória principal

Professor: Wellington Franco

Introdução

> *A **memória principal (RAM)** é um recurso importante que deve ser gerenciado com muito cuidado. Apesar dos computadores atuais possuírem memórias dez mil vezes maiores do que o IBM 7094, os programas tornam-se maiores muito mais rapidamente do que as memórias. Pode-se afirmar que "programas tendem a se expandir a fim de ocupar toda a memória disponível".*
>
> **TANENBAUM, A. (2010)**

Organização hierárquica de memórias



Acessíveis pelo processador. Os dados/instruções precisam estar aqui para serem acessadas.

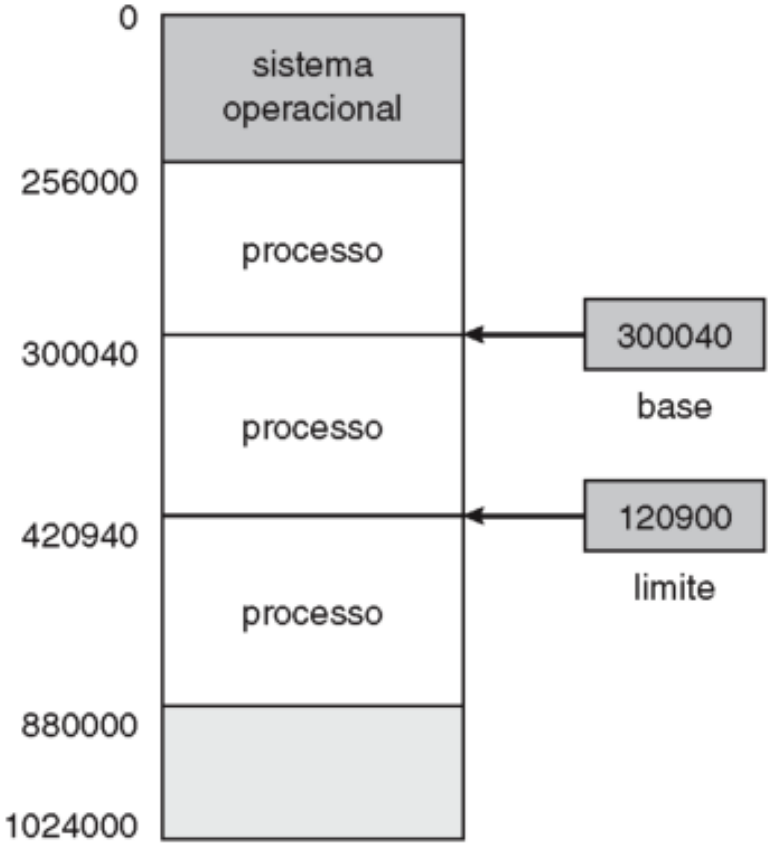
Hardware básico

- Os programas precisam ser trazidos para a memória principal para poderem ser executados (criação de um processo)
- Memória principal e registrados são as únicas unidades de armazenamento acessadas diretamente pela CPU
- O acesso a um registrador é feito em um (ou menos) ciclos de CPU
- Acessos à memória principal podem levar mais ciclos
- A memória cache se situa entre a memória principal e os registradores

Hardware básico

- Mecanismos de proteção de memória são necessários para garantir a operação correta
  - Um par de registradores definem o espaço de endereçamento lógico
    - **Registrador base**
      - Contém o menor endereço válido da memória física
    - **Registrador limite**
      - Especifica o tamanho do intervalo

Hardware básico



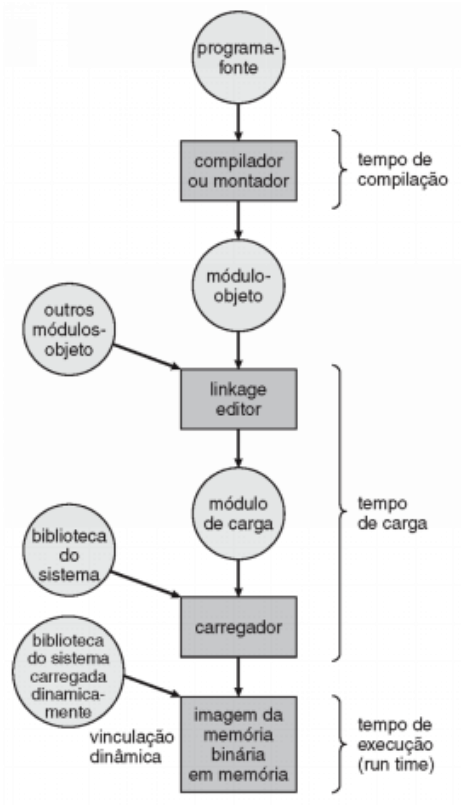
Vinculação de endereços

- **Tempo de compilação**: Se a localização de memória for conhecida a priori, o compilador pode gerar código absoluto; requer recompilação caso a localização mude

- **Tempo de carga**: Requer a geração de código relocável se a localização de memória não for conhecida em tempo de compilação

- **Tempo de execução**: A vinculação é adiada até o momento da execução caso o processo possa ser movido de um segmento de memória para outro durante a execução.

Vinculação de endereços



Espaço de endereçamento lógico x físico

- **Endereço lógico**: gerado pela CPU, também conhecido como endereço virtual

- **Endereço físico**: endereço visto pela unidade de memória

- **Espaço de endereçamento lógico:** conjunto de todos os endereços lógicos gerados por um programa
- **Espaço de endereçamento físico**: os endereços físicos correspondentes a esses endereços lógicos.

- Endereços lógicos e físicos são os mesmos em esquemas de vinculação de endereços em tempo de compilação ou tempo de carga e são diferentes em esquemas de vinculação de endereços em tempo de execução

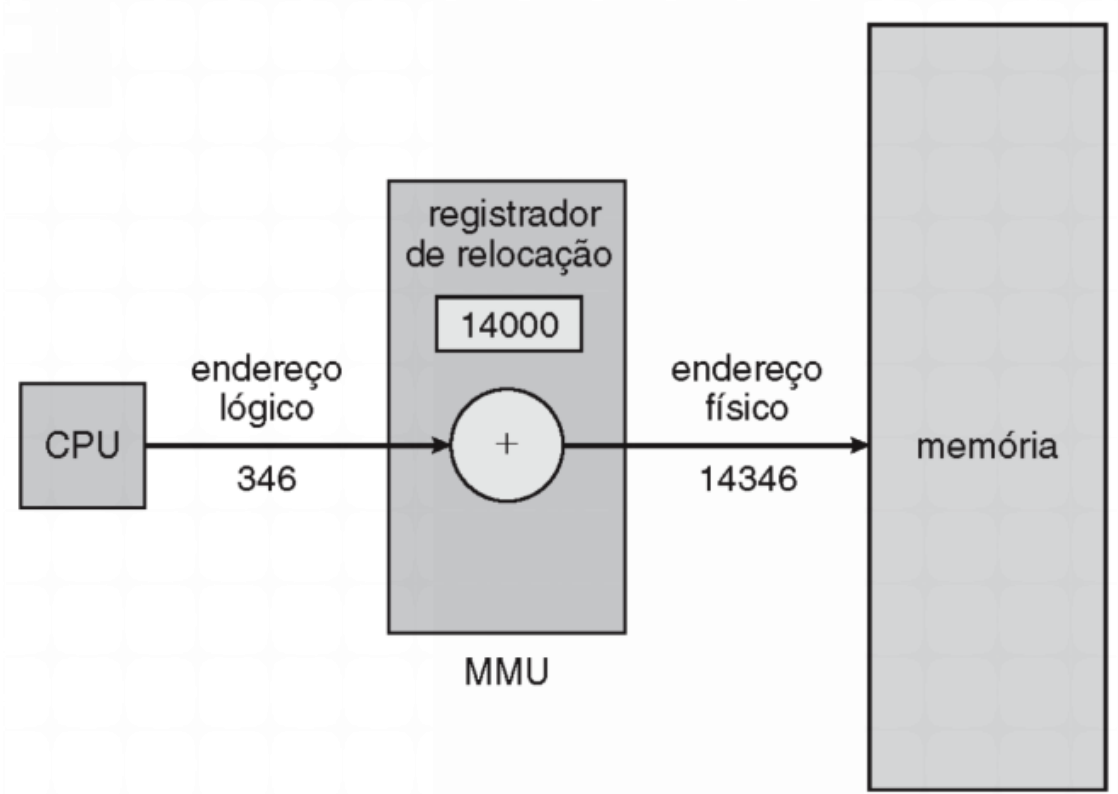
Unidade de Gerenciamento de Memória (MMU)

- Dispositivo de hardware que mapeia um endereço virtual em um endereço físico

- Na MMU, o valor de um registrador de relocação é somado a todo endereço gerado por um processo do usuário no momento em que ele é enviado a memória

- O programa lida apenas com endereços lógicos; ele nunca vê os endereços físicos reais

Unidade de Gerenciamento de Memória (MMU)



Carga dinâmica

- **Programa inteiro carregado na memória -> limitação da memória**
- Uma rotina não é carregada até ser invocada
- Melhor utilização do espaço de memória; rotinas não utilizadas nunca são carregadas
- Útil quando grandes quantidade de código são necessárias para lidar com casos infrequentes
- Não requer nenhum suporte especial do sistema operacional

Troca de processos (swapping)

> *A estratégia mais simples, chamada troca de processos (swapping), consiste em trazer, em sua totalidade, cada processo para a memória, executá-lo durante um certo tempo e, então, devolvê-lo no disco, de forma que não ocupem qualquer espaço na memória quando não estão executando.*
>
> **TANENBAUM, A. (2010)**

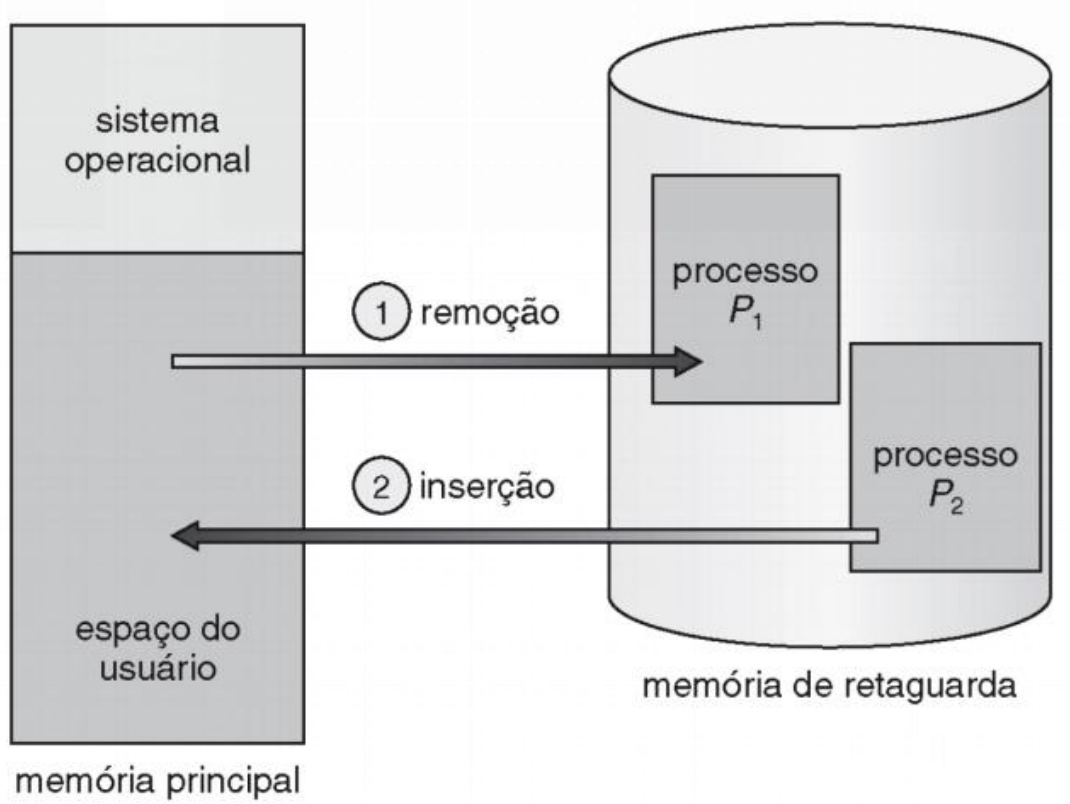
Swapping

- Um processo pode ser removido temporariamente da memória para um armazenamento secundário e depois trazido de volta para continuar sua execução

- Roll out, Roll in – variante de swapping, utilizada em algoritmos de escalonamento com prioridade; processos de baixa prioridade são removidos para que processos de mais alta prioridade possam ser carregados e executados

- A maior parte do tempo de swap é gasto transferindo dados

Swapping


sistema operacional
espaço do usuário
memória principal
1 remoção
2 inserção
processo P_1
processo P_2
memória de retaguarda

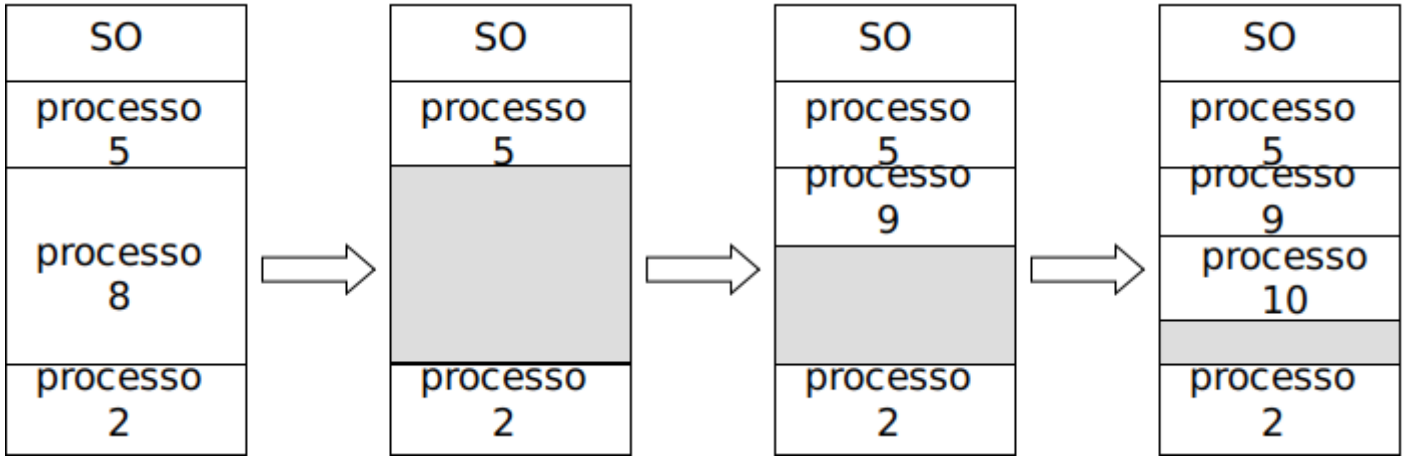
Alocação contínua

- A memória principal é geralmente dividida em duas partições:
  - Sistema operacional residente, geralmente na memória baixa, junto com o vetor de interrupções
  - Processos do usuário na parte alta da memória
- Registradores de relocação são utilizados para proteger os processos dos usuários uns dos outros e também para evitar alterações em dados e códigos do sistema operacional
- O registrador base contém o menor endereço de memória permitido
- O registrador de limite define o tamanho do espaço de endereçamento – cada endereço lógico deve ser menor que o registrador de limite
- A MMU mapeia os endereços lógicos dinamicamente

Alocação contínua

- Alocação com múltiplas partições
- **Buraco:** bloco de memória disponível; buracos de vários tamanhos são espalhados na memória
- Quando um processo é criado ele é alocado em um buraco grande o suficiente para suas necessidades de memória
- O sistema operacional mantem informações sobre:
  a) partições alocadas b) partições livres (buracos)

Alocação contínua



Alocação contínua

- **Problema na alocação dinâmica de memória**
  - Como atender uma solicitação de tamanho **n** a partir de uma lista de intervalos livres.

  - **First-fit**: Aloca o primeiro buraco que for grande o suficiente
  - **Best-fit**: Alocar o menor buraco que for grande o suficiente; precisa pesquisar a lista inteira a menos que esteja ordenada por tamanho
    - Produz o menor buraco remanescente
  - **Worst-fit**: Aloca o maior buraco; também precisa percorrer toda a lista
    - Produz o maior buraco remanescente

Fragmentação

- **Fragmentação externa:** exige memória suficiente para atender uma requisição mas ela não é contínua

- **Fragmentação interna:** a memória alocada pode ser levemente maior que a memória requisitada; esta diferença de memória faz parte da partição mas não é utilizada

- Reduzir fragmentação utilizando **compactação**
  - Ordena o conteúdo da memória em uma localidade para juntar todo o espaço livre em um grande bloco
  - Compactação só é possível com relocação dinâmica

Paginação

- O espaço de endereçamento físico de um processo pode ser não-contínuo
- A memória física é dividida em blocos de tamanho fixo chamados **quadros**
- A memória lógica é dividida também em blocos do mesmo tamanho chamados **páginas**
- Para executar um programa de **n** páginas é preciso encontrar **n** quadros para carregar o programa

Paginação

- Uma tabela de paginação é utilizada para traduzir endereços lógicos em endereços físicos

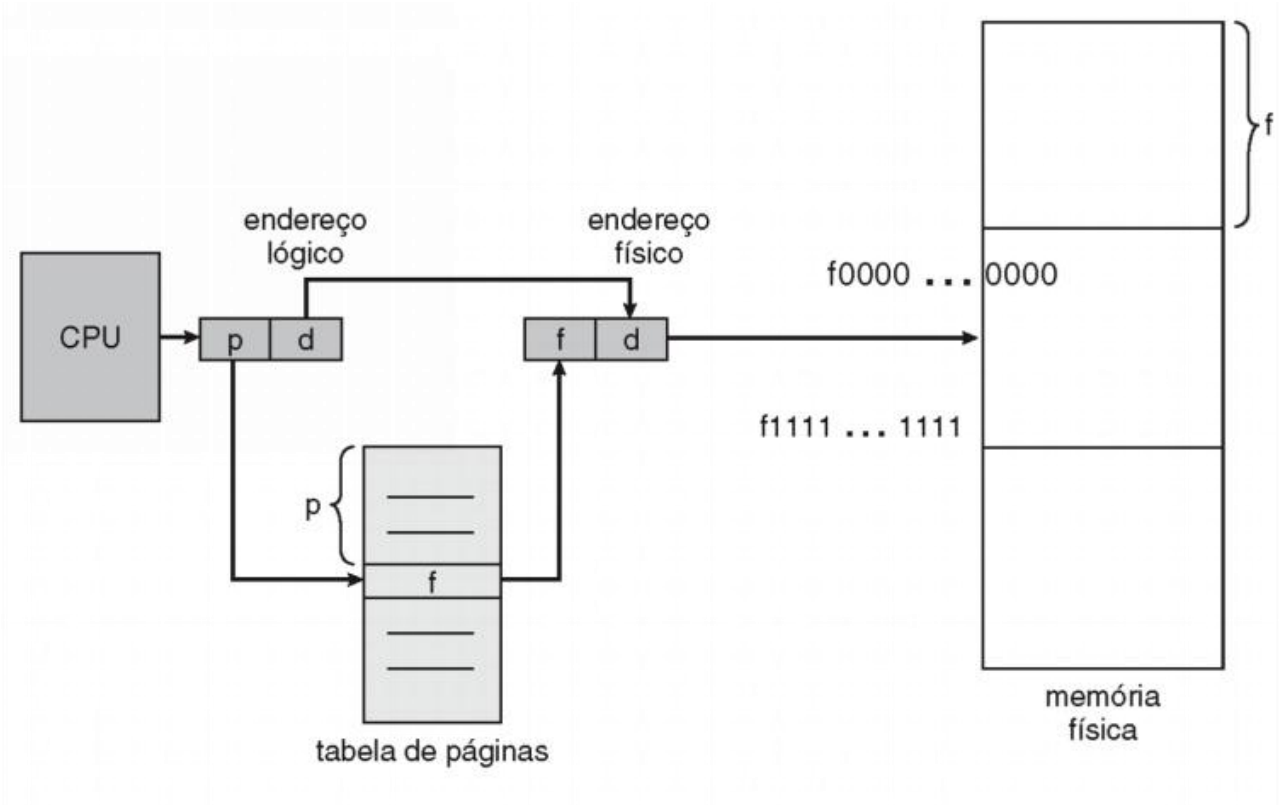
- Os endereços gerados pela CPU são divididos em duas partes:

    - **Número da página (p)** – utilizado como índice na tabela de paginação que contém o endereço base de cada página na memória física
    - **Deslocamento (d)** – combinado com o endereço base define o endereço físico a ser enviado para a unidade de memória

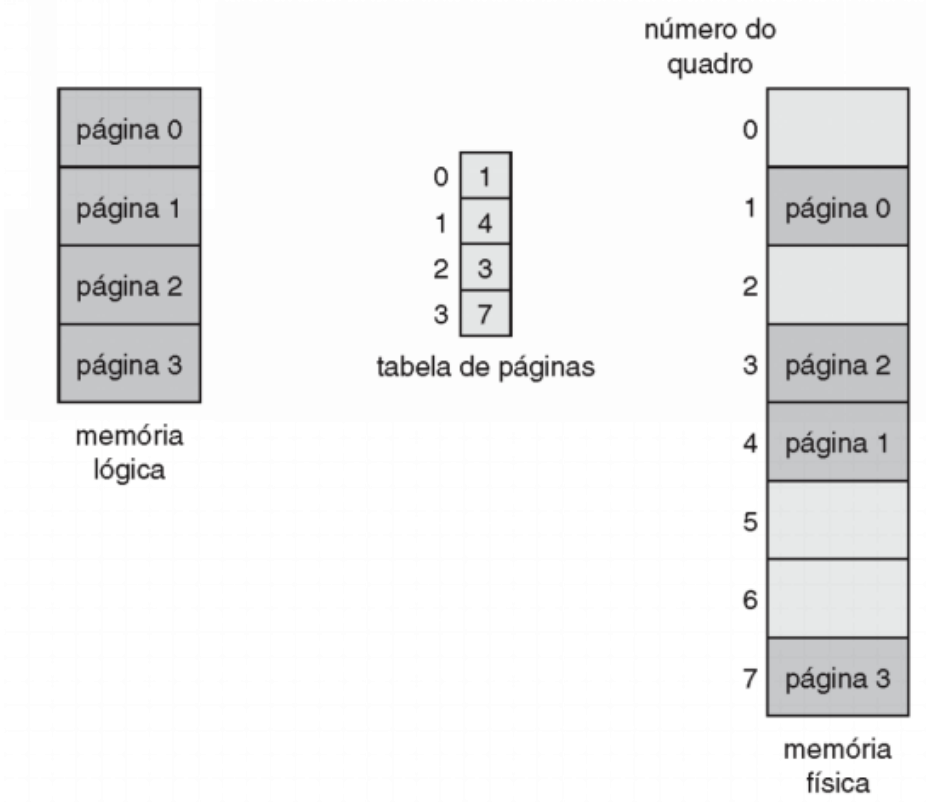
Paginação

- Para um determinado espaço de endereçamento com 2^m bits e páginas de tamanho 2^n

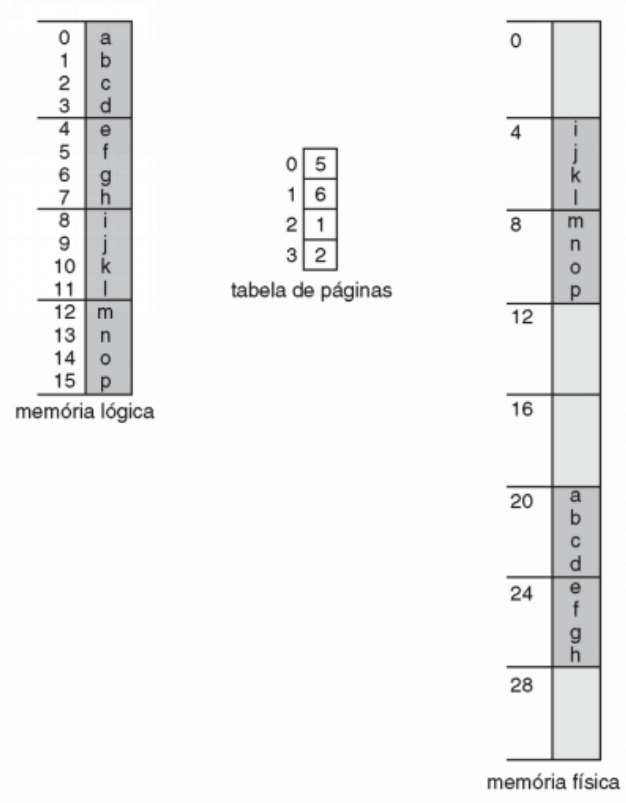
Hardware de paginação



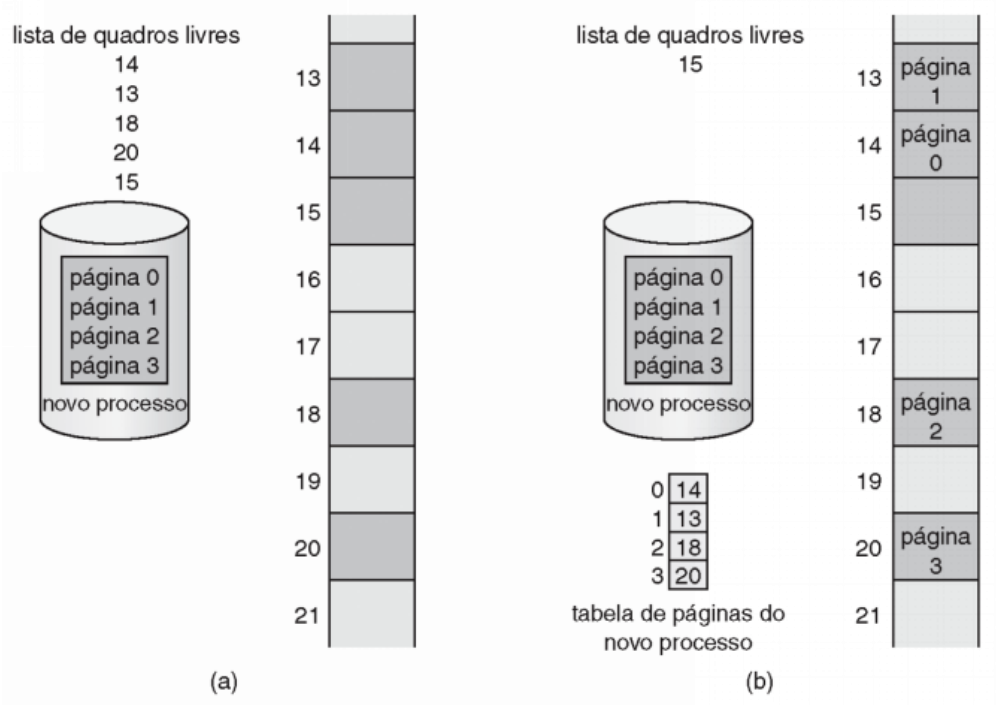
Modelo de paginação



Exemplo de paginação



Quadros livres



antes da alocação

depois da alocação

Paginação

- Uma vantagem da paginação é a possibilidade de compartilhamento de código comum
  - Exemplo: 40 usuários utilizando um editor de texto (150KB) e 50 KB de espaço de dados
    - Total = 40 * (150 + 50) = 8000 KB

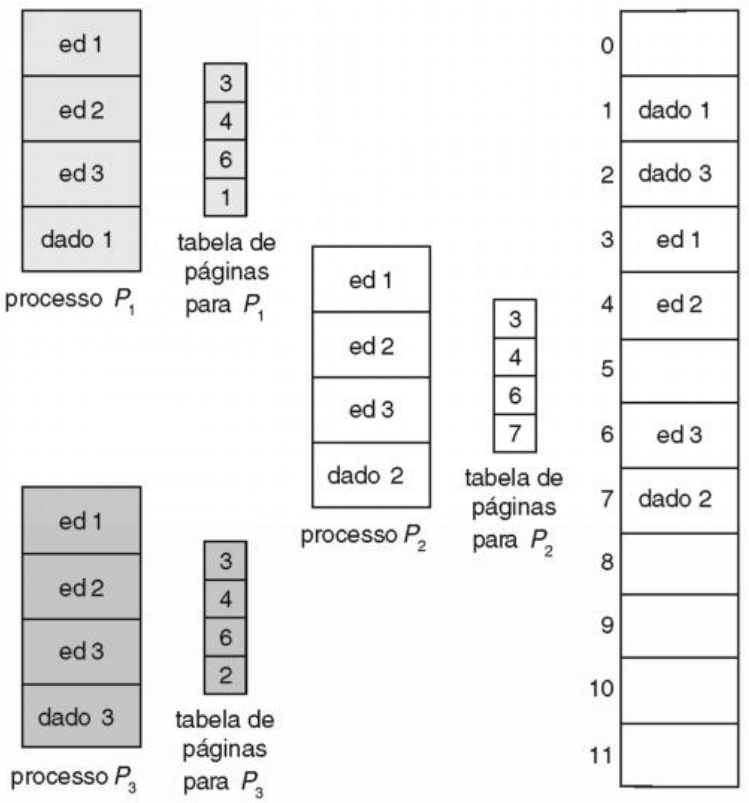
- Código compartilhado
  - Uma cópia apenas de leitura do código compartilhada entre vários processos
  - O código compartilhado precisa aparecer no mesmo local no espaço de endereçamento lógico de todos os processos

Paginação

- Código e dados privados
  - Cada processo mantém uma cópia separada dos dados e código
  - As páginas para o código privado podem aparecer em qualquer lugar no espaço de endereçamento lógico

- Exemplo:
  - Cai para 40 * 50 + 150 = 2150 KB

Paginação


ed 1
ed 2
ed 3
dado 1
processo P_1
3
4
6
1
tabela de páginas para P_1
ed 1
ed 2
ed 3
dado 2
processo P_2
3
4
6
7
tabela de páginas para P_2
ed 1
ed 2
ed 3
dado 3
processo P_3
3
4
6
2
tabela de páginas para P_3
0
1 dado 1
2 dado 3
3 ed 1
4 ed 2
5
6 ed 3
7 dado 2
8
9
10
11

Estrutura da tabela de páginas

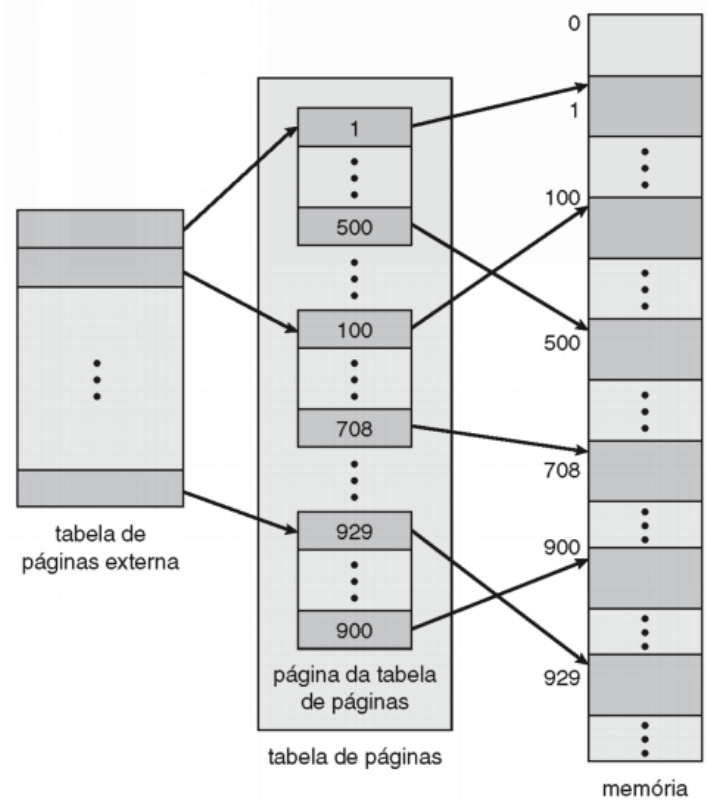
- **Paginação hierárquica**
- **Tabelas de paginação com hash**
- **Tabela de paginação invertida**

Paginação hierárquica

- Quebra o espaço de endereçamento lógico em múltiplas tabelas de paginação

- Uma técnica simples é a paginação em dois níveis

Paginação hierárquica



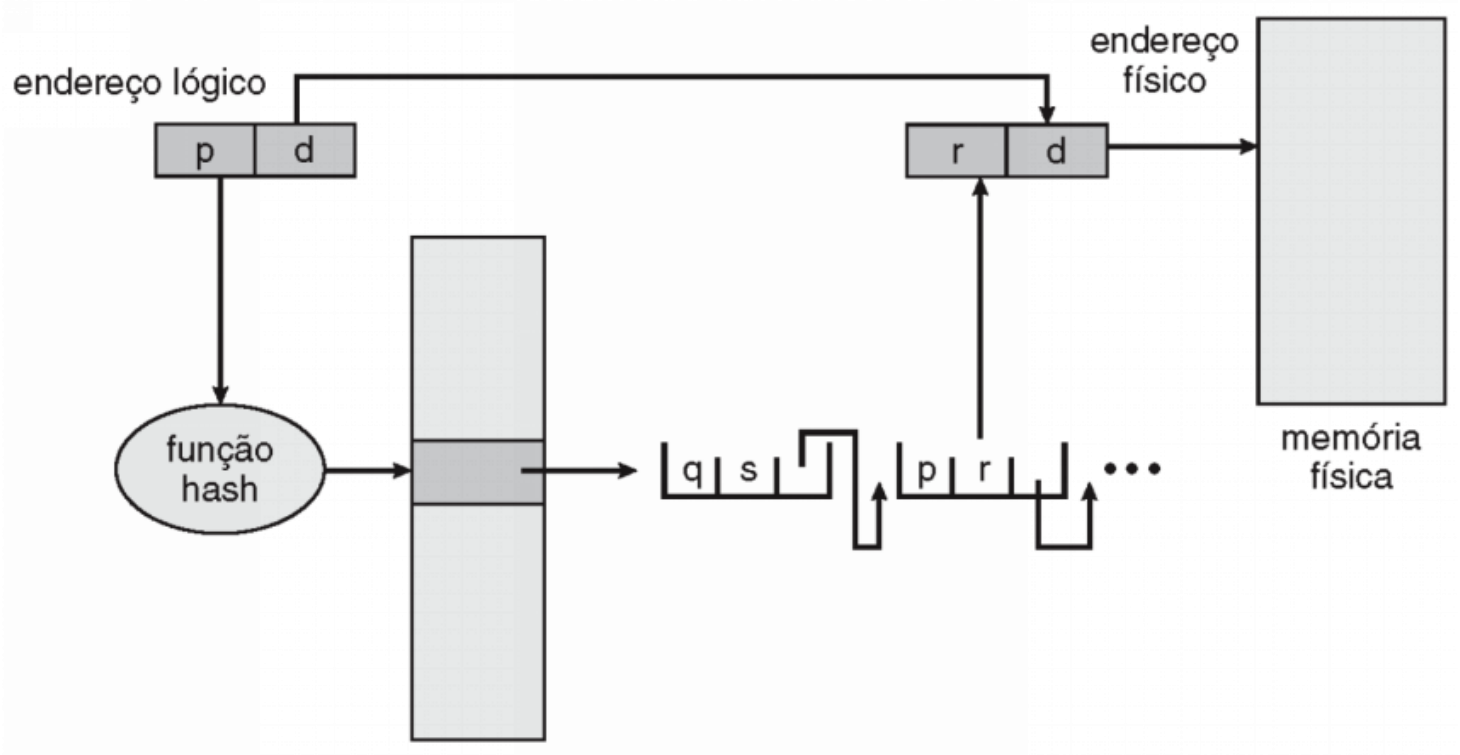
Tabelas de paginação com hash

- Comum em espaço de endereçamento com mais de 32 bits

- O número da página virtual é submetido a uma função hash e o resultado aponta para uma entrada na tabela de paginação
  - Cada entrada na tabela de paginação contém uma lista de elementos cujo hash apontou para a mesma localidade (colisões)

- O número da página virtual é então comparado com os valores nessa lista para localizar o quadro na memória física

Tabelas de paginação com hash

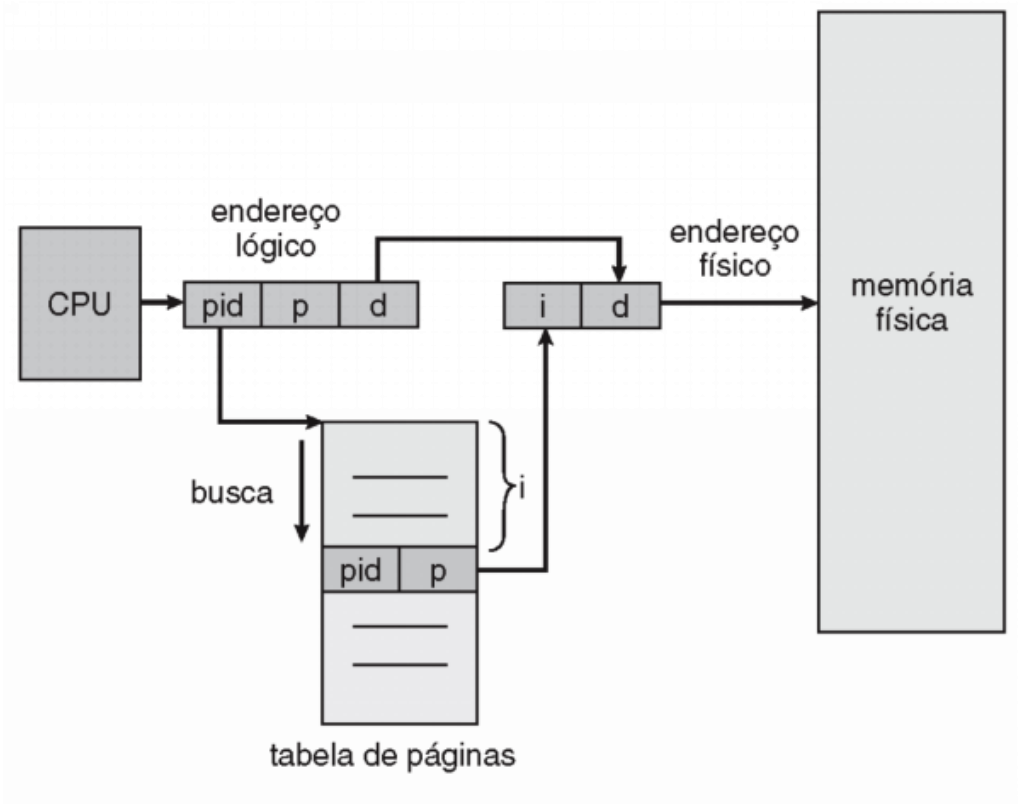


Tabelas de paginação invertida

- Uma entrada para cada quadro na memória principal

- As entradas da tabela são os endereços das páginas virtuais armazenadas nessas localização de memória, com informação sobre o processo proprietário

Tabelas de paginação invertida



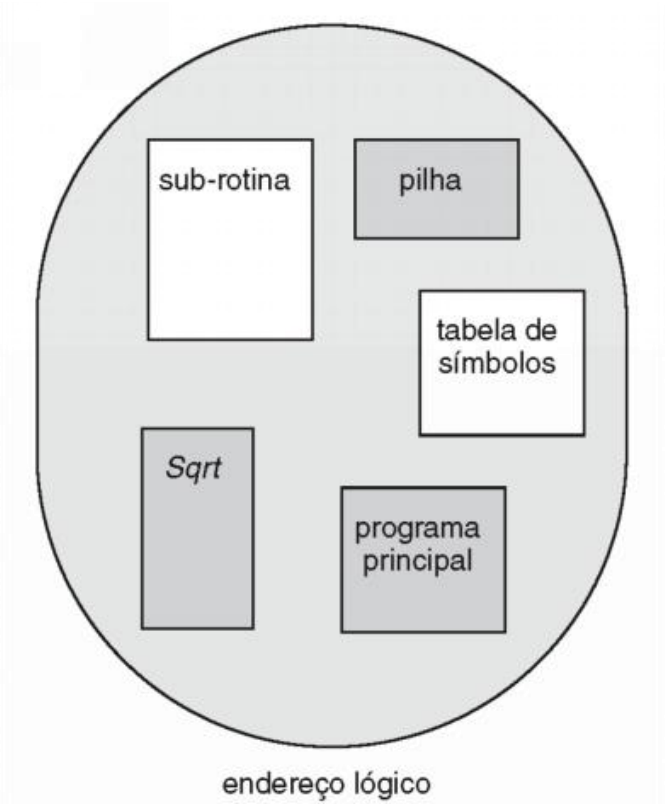
Segmentação

- Esquema de gerência de memória que reflete a visão que o usuário tem da memória

- Um programa é uma coleção de segmentos

    - Um segmento representa uma unidade lógica, como:
        - Programa principal
        - Procedimento/Função/Método
        - Objeto/Variáveis locais/Variáveis globais
        - Pilha
        - Tabela de símbolos

Segmentação



Segmentação

- O espaço de endereçamento lógico consiste em uma tupla:

  <número do segmento, deslocamento>,

- Tabela de segmentos – mapeia os endereços virtuais em endereços físicos; cada entrada tem:
  - **base** – contém o endereço inicial do segmento na memória
  - **limite** – especifica o tamanho do segmento

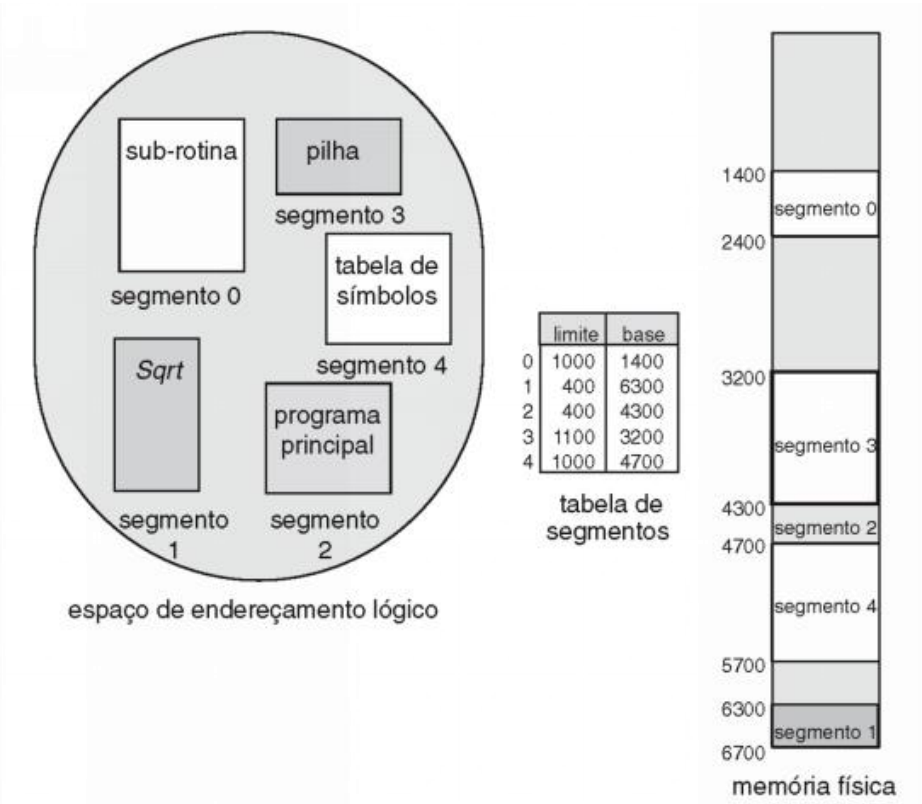
Segmentação

- **Segment-table Base Register (STBR)** aponta para a base da tabela de segmentação na memória

- **Segment-table Length Register (STLR)** indica o número de segmentos utilizados por um programa;

  o número de segmento **S** é legal se **S < STLR**

Exemplo de uso de segmentação



Dúvidas??

E-mail: wellington@crateus.ufc.br