DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

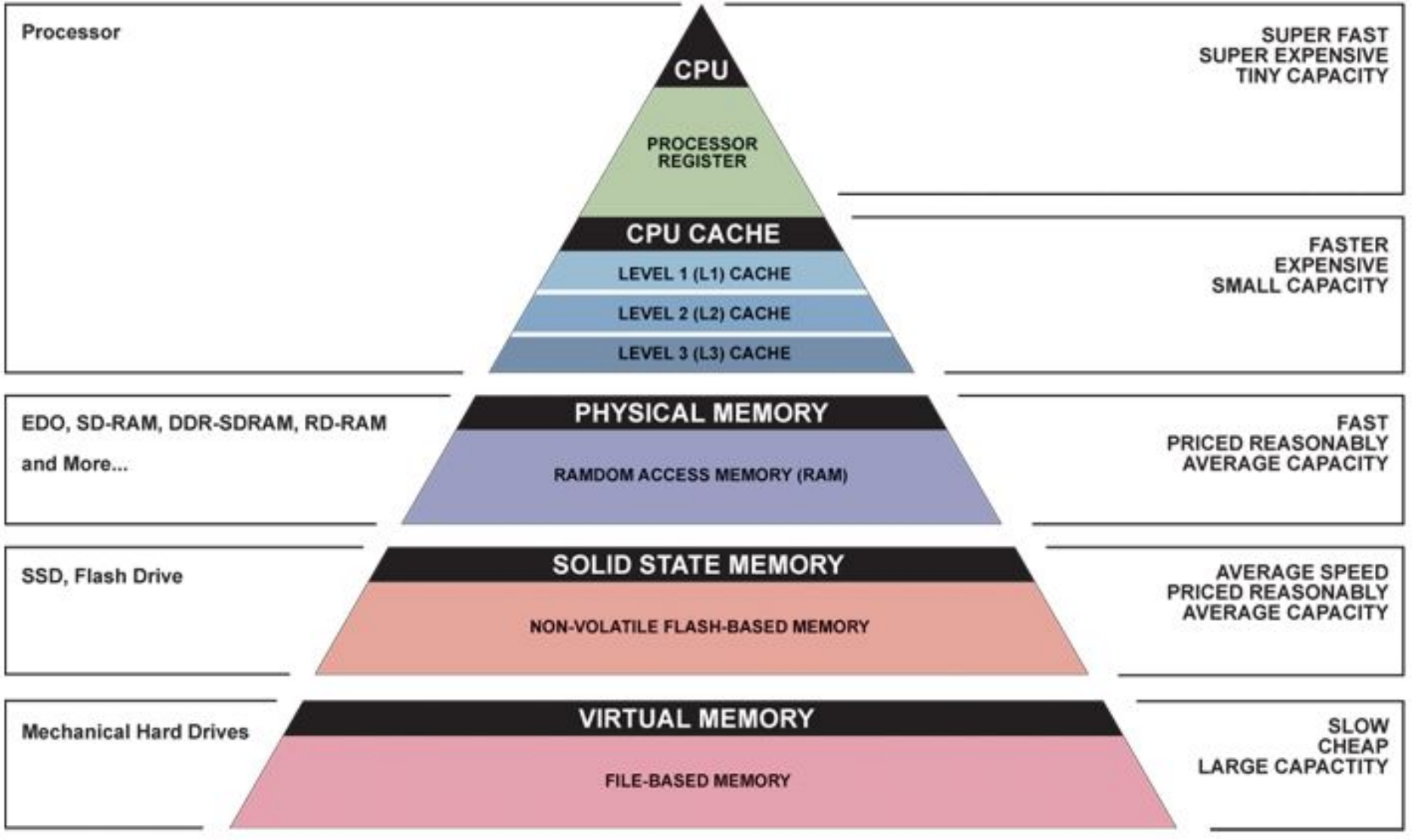
UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Estruturas de Dados

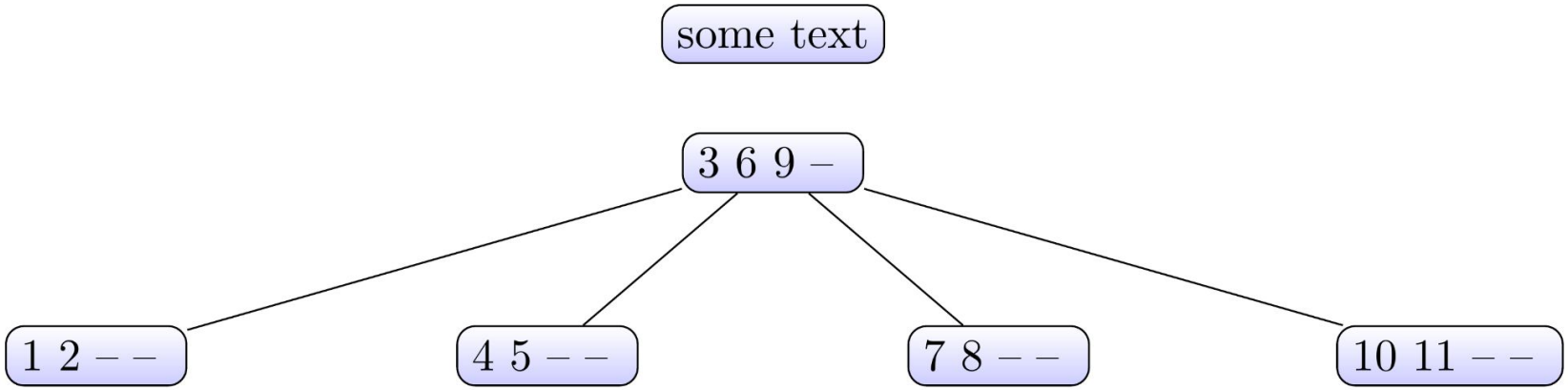
Memória Secundária

Professores: Anisio Lacerda
Lucas Ferreira
Wagner Meira Jr.
Washington Cunha

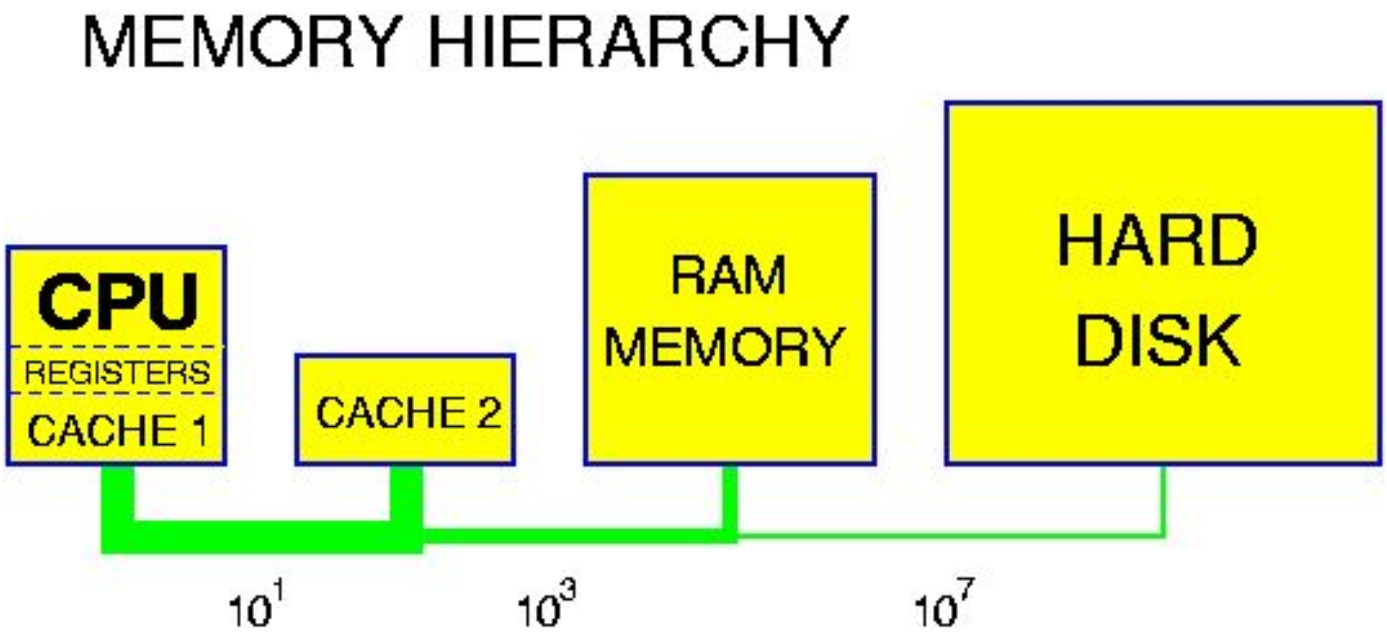
Hierarquia de Memória



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng



Hierarquia de Memória



Número aproximado de ciclos de CPU para acessar diferentes elementos da hierarquia de memória

Memória Interna x Memória Externa

- Os algoritmos vistos até aqui consideram que os dados podem ser armazenados na memória interna (física) do computador
  - Custo dos algoritmos é relacionado com o número de comparações e movimentações

- O que fazer quando a quantidade de dados é maior que o tamanho da memória interna?

Memória Interna x Memória Externa

- Nesse caso, devemos **minimizar o número de acessos à memória externa**, que têm um custo muito maior que comparações e movimentações de registros.
- É necessário trabalhar com “blocos” de dados, trazendo-os para a memória principal quando necessário e normalmente o acesso aos dados é sequencial
- Um conceito importante para isso é a **Localidade de Referência**

Analogia - Biblioteca

- Antes de utilizar um livro, você o procura na estante (memória externa), e o carrega até uma mesa (memória interna)
- Custo para achar o livro na estante é maior do que pegá-lo da mesa

“Memória externa”

“Memória interna”

- Vários livros para consultar: carregar um “bloco” de livros de uma vez para a mesa pode ser vantajoso

Localidade de Referência

- Padrões de acesso a dados observados desde os primeiros sistemas de computação:
  - **Temporal**: se um dado é acessado uma vez, há uma probabilidade grande dele ser acessado novamente num futuro próximo.
  - **Espacial**: se um dado é acessado uma vez, há uma probabilidade grande do seu vizinho ser acessado em breve
- A localidade de referência permite que os algoritmos possam trabalhar de forma eficiente utilizando blocos de dados

Localidade de Referência

- Exemplo: Loops

```
for (i = 0; i < n; i++) {
        vetor[i] = fun(vetor[i]);
}
```

- Localidade de referência espacial:
  - vetor[0], vetor[1], …, vetor[n-1]

- Localidade de referência temporal:
  - Acesso às instruções compondo o corpo do loop

O Desafio do Armazenamento Secundário

- **Estruturas de dados para armazenamento secundário:**

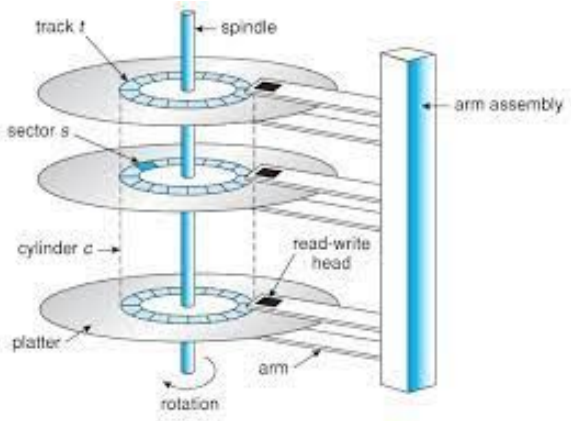
  Estruturas de Dados para memória primária, projetadas para acesso aleatório na memória, não são eficientes para os padrões de endereçamento físico e acesso sequencial de dispositivos de armazenamento secundário.
- **Endereçamento físico:**

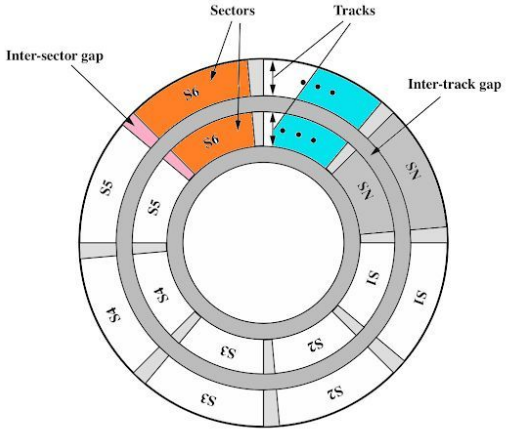
  Os dados são armazenados em blocos de tamanho fixo no dispositivo de armazenamento. Estruturas de dados tradicionais podem exigir a colocação de dados dispersos em vários blocos, impactando a velocidade de acesso.
- **Acesso sequencial:**

  A recuperação de dados é mais rápida quando os blocos são contíguos. O armazenamento secundário é otimizado para acesso sequencial de dados, enquanto a RAM permite acesso aleatório.

Disco Magnético

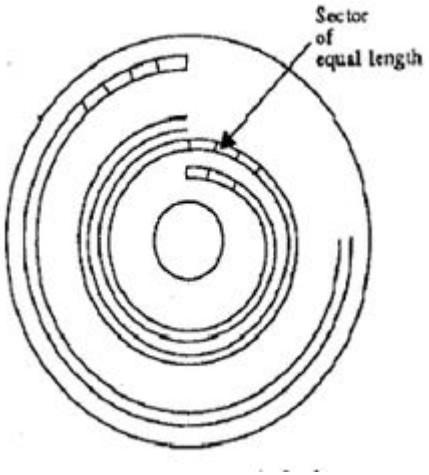
- Dividido em círculos concêntricos (trilhas).
- Cilindro → todas as trilhas verticalmente alinhadas e que possuem o mesmo diâmetro.
- Latência rotacional → tempo necessário para que o início do bloco contendo o registro a ser lido passe pela cabeça de leitura/gravação.
- Tempo de busca (seek time) → tempo necessário para que o mecanismo de acesso desloque de uma trilha para outra (maior parte do custo para acessar dados).
- Acesso seqüencial indexado = acesso indexado + organização seqüencial,
- Aproveitando características do disco magnético e procurando minimizar o número de deslocamentos do mecanismo de acesso → esquema de índices de cilindros e de páginas.



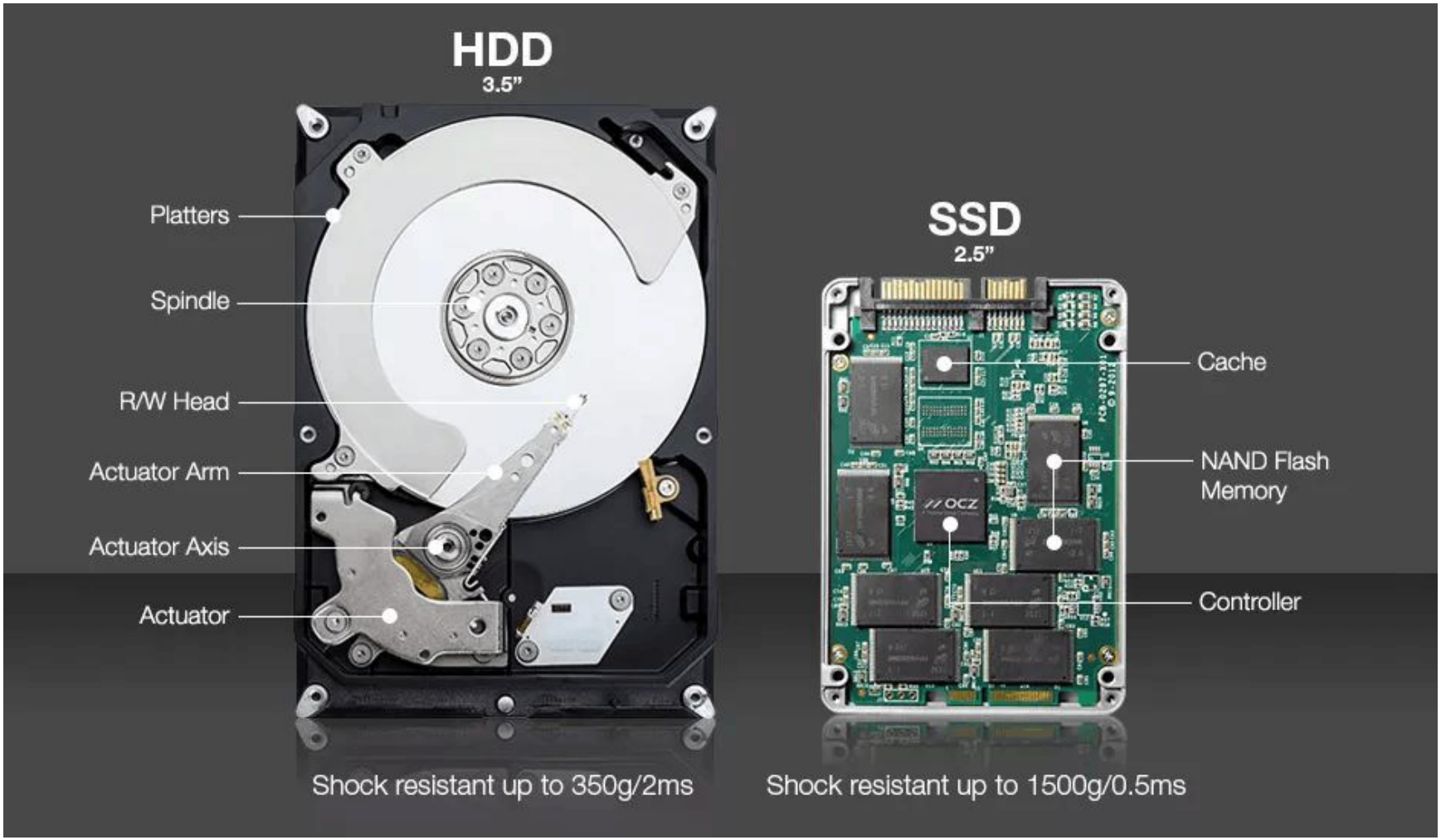


Discos Óticos de Apenas-Leitura (CD-ROM)

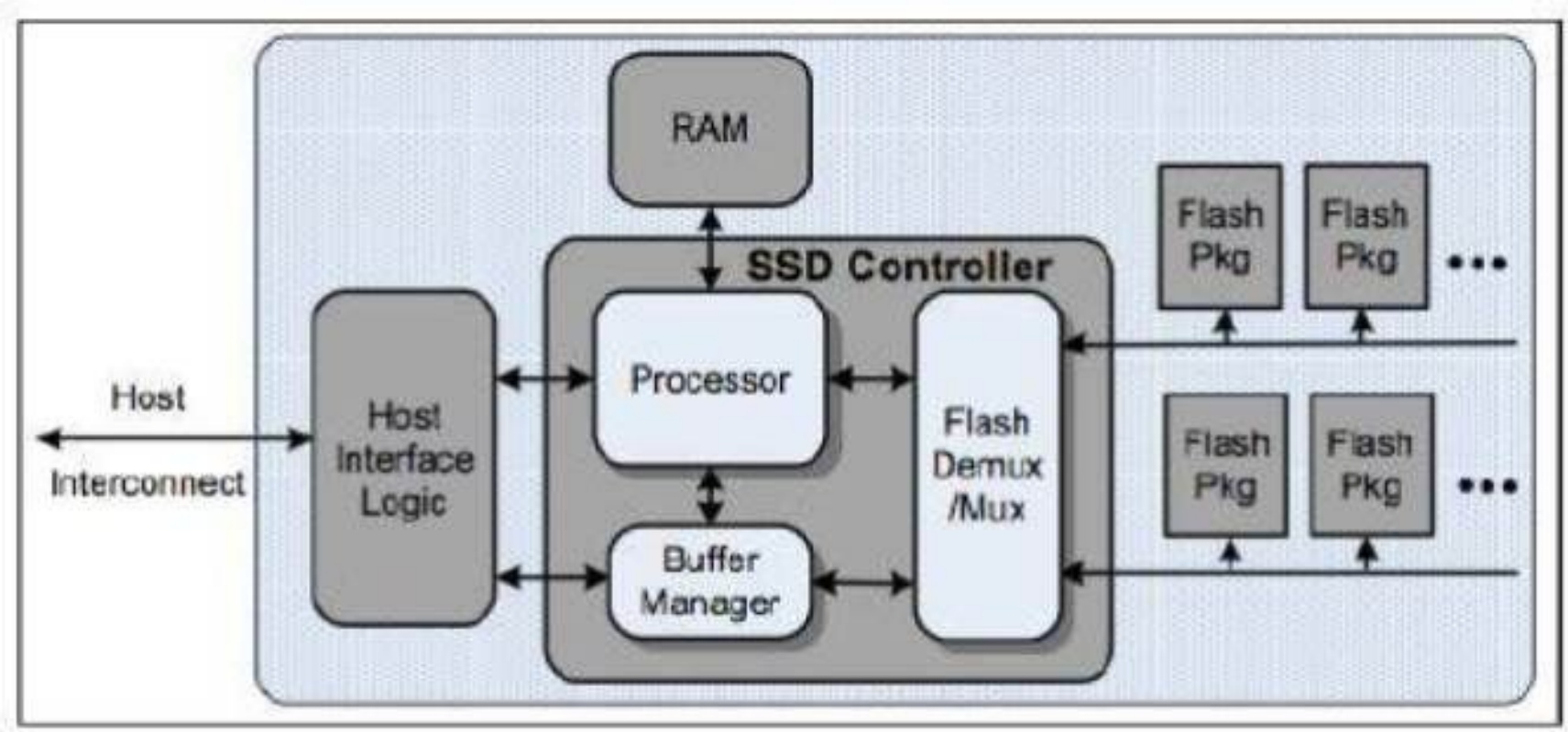
- Grande capacidade (para a época) de armazenamento (600 MB) e baixo custo.
- Informação armazenada é estática.
- A eficiência na recuperação dos dados é afetada pela localização dos dados no disco e pela seqüência com que são acessados.
- Velocidade linear constante → trilhas possuem capacidade variável e tempo de latência rotacional varia de trilha para trilha.
- A trilha tem forma de uma espiral contínua.
- Tempo de busca: acesso a trilhas mais distantes demanda mais tempo que no disco magnético. Há necessidade de deslocamento do mecanismo de acesso e mudanças na rotação do disco.
- Varredura estática: acessa conjunto de trilhas vizinhas sem deslocar mecanismo de leitura.
- Estrutura seqüencial implementada mantendo-se um índice de cilindros na memória principal.



Solid State Drives



Solid State Drives



SSD Logic Components

Sistema de Arquivos

- **Arquivos e diretórios em uma estrutura hierárquica:** Cria um sistema de pastas para organização eficiente de dados, semelhante a um catálogo de biblioteca.
- **Atributos, localização e permissões de acesso:** Armazena informações essenciais sobre cada arquivo, incluindo tamanho, data de criação e quem pode acessá-lo.
- **Sistemas de arquivos (FAT, NTFS, UNIX):**

  Diferentes sistemas de arquivos usam técnicas variadas para gerenciar a alocação de arquivos (onde os dados reais de cada arquivo estão localizados no dispositivo de armazenamento).

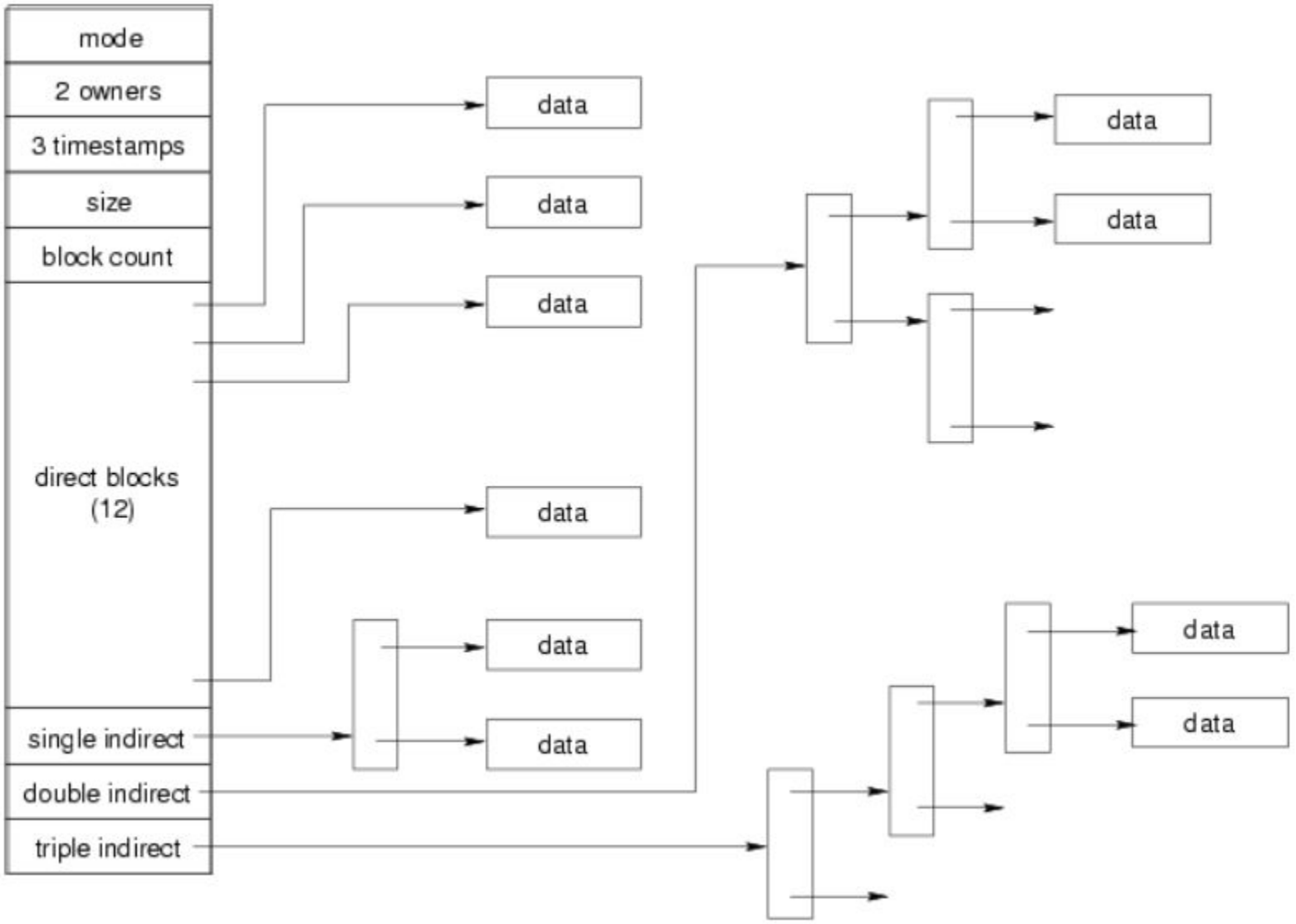
Sistema de Arquivos

O sistema de arquivos é a estrutura fundamental para a memória secundária. Ele organiza arquivos e diretórios em uma hierarquia, rastreia atributos dos arquivos (tamanho, data de criação, permissões) e gerencia a alocação de blocos de armazenamento para os dados dos arquivos.

- **Busca:** O sistema de arquivos permite localizar um arquivo pelo nome, navegando pela hierarquia de diretórios e consultando a tabela de alocação de arquivos para encontrar os blocos onde o arquivo está armazenado.
- **Inserção:** Envolve criar uma nova entrada no diretório pai e alocar blocos livres no dispositivo de armazenamento para armazenar os dados do novo arquivo.
- **Remoção:** Localiza o arquivo a ser excluído, libera os blocos associados ao arquivo e remove a entrada do diretório pai.

Localidade de referência: O sistema de arquivos permite agrupar blocos contíguos para um mesmo arquivo, explorando a localidade de referência para leituras e escritas sequenciais.

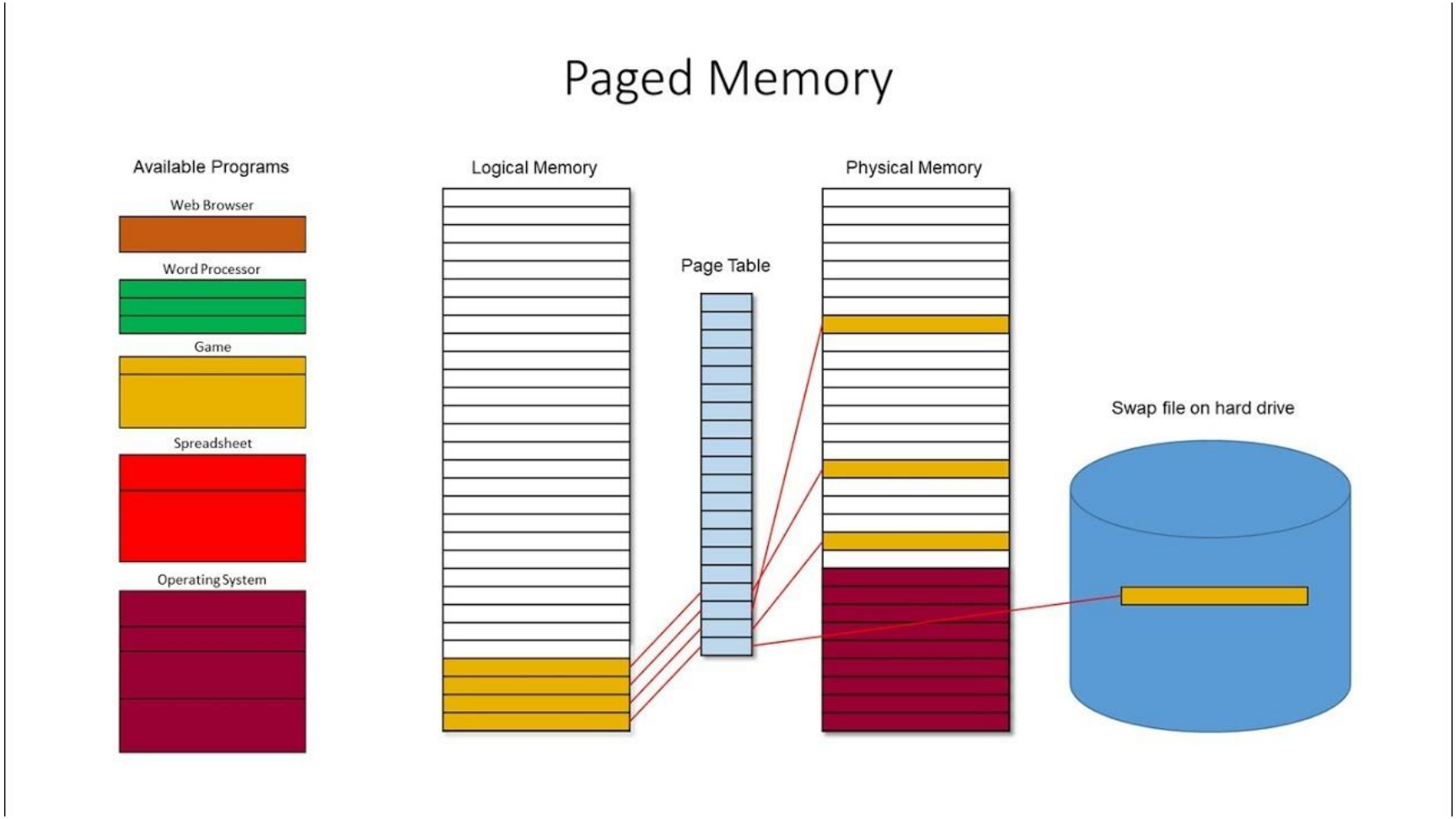
Sistema de Arquivos



Pesquisa em Memória Secundária

- Pesquisa em memória secundária: arquivos contém mais registros do que a memória interna pode armazenar.
- Medida de complexidade: custo de transferir dados entre a memória principal e secundária (minimizar o número de transferências).
  - Custo para acessar um registro é algumas ordens de grandeza maior do que o custo de processamento na memória primária.
  - Memórias secundárias: apenas um registro pode ser acessado em um dado momento (acesso seqüencial).
  - Memórias primárias: acesso a qualquer registro de um arquivo a um custo uniforme (acesso direto).
- O aspecto sistema de computação é importante.
  - As características da arquitetura e do sistema operacional da máquina tornam os métodos de pesquisa dependentes de parâmetros que afetam seus desempenhos

Paginação de Memória



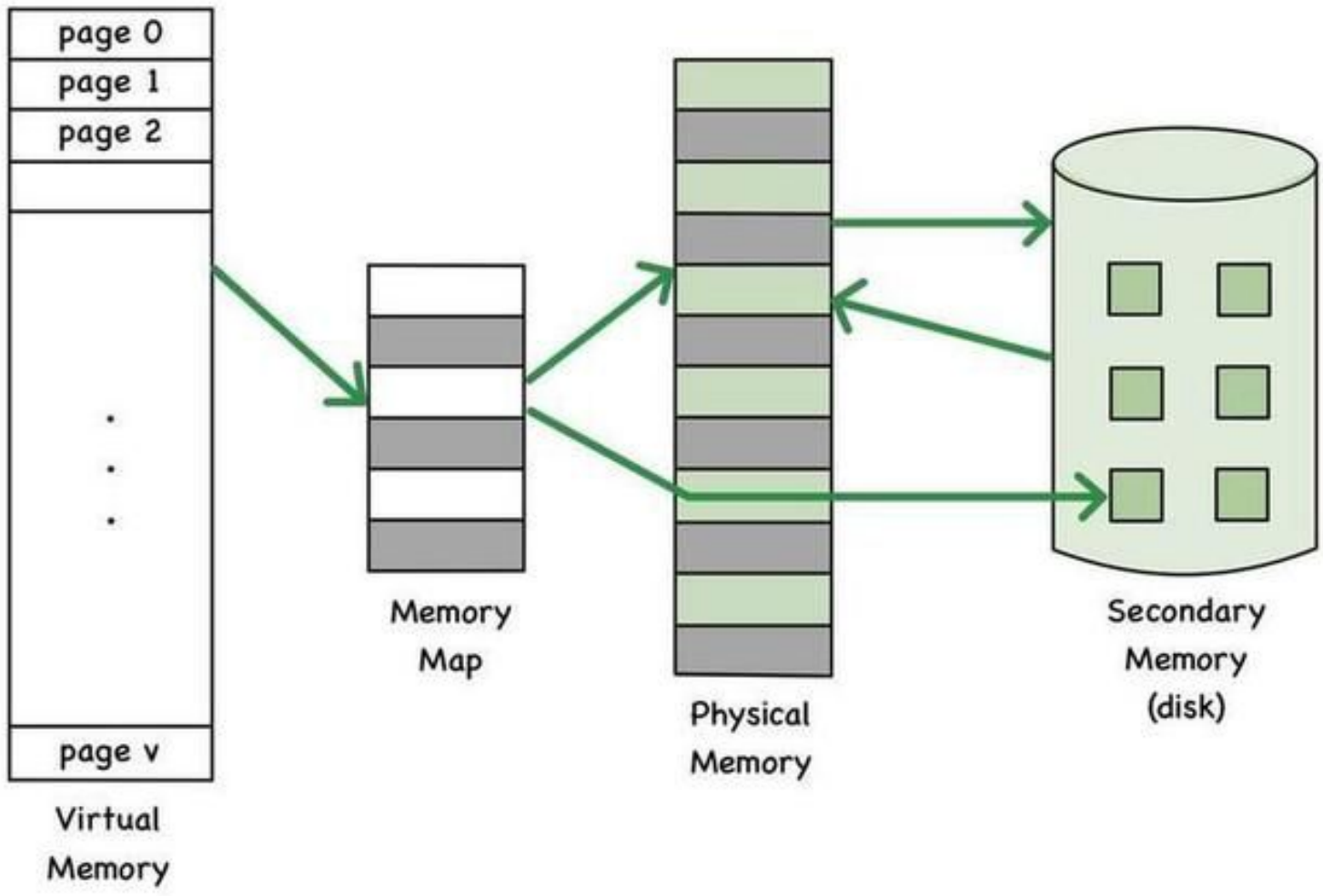
Memória Virtual

- Normalmente implementado como uma função do sistema operacional.
- Modelo de armazenamento em dois níveis, devido à necessidade de grandes quantidades de memória e o alto custo da memória principal.
- Uso de uma pequena quantidade de memória principal e uma grande quantidade de memória secundária.
- Programador pode endereçar grandes quantidades de dados, deixando para o sistema a responsabilidade de transferir o dado da memória secundária para a principal.
- Boa estratégia para algoritmos com localidade de referência.
- Organização do fluxo entre a memória principal e secundária é extremamente importante.

Memória Virtual

- Organização de fluxo → transformar o endereço usado pelo programador na localização física de memória correspondente.
  - Espaço de Endereçamento → endereços usados pelo programador.
  - Espaço de Memória → localizações de memória no computador.
- O espaço de endereçamento N e o espaço de memória M podem ser vistos como um mapeamento de endereços do tipo: f : N → M .
- O mapeamento permite ao programador usar um espaço de endereçamento que pode ser maior que o espaço de memória primária disponível.

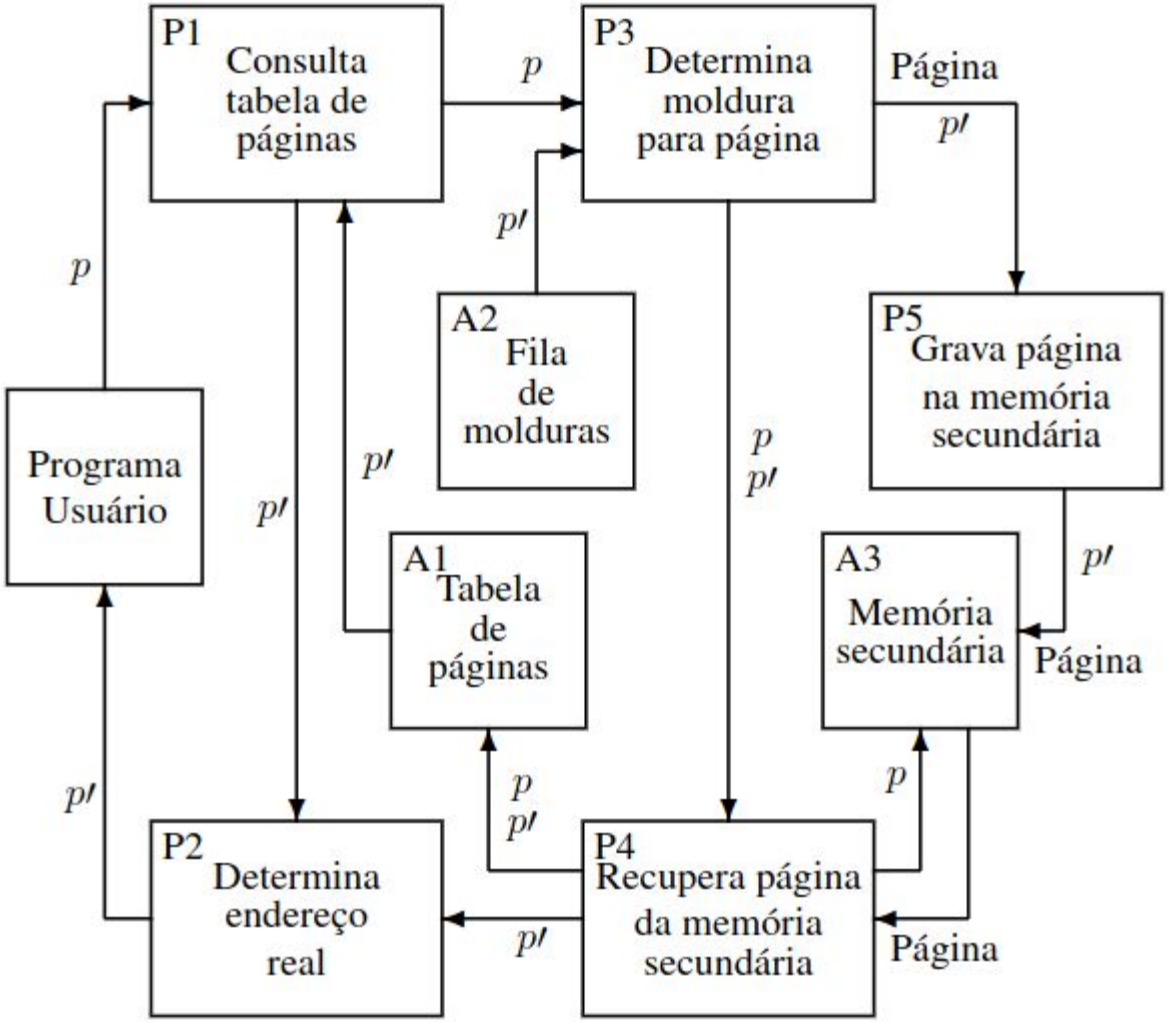
Memória Virtual



Memória Virtual

- O espaço de endereçamento é dividido em páginas de tamanho igual em geral, múltiplos de 4 Kbytes.
- A memória principal é dividida em molduras de páginas de tamanho igual.
- As molduras de páginas contêm algumas páginas ativas enquanto o restante das páginas estão residentes em memória secundária (páginas inativas).
- O mecanismo possui duas funções:
  - Mapeamento de endereços → determinar qual página um programa está endereçando, encontrar a moldura, se existir, que contenha a página.
  - Transferência de páginas → transferir páginas da memória secundária para a memória primária e transferí-las de volta para a memória secundária quando não estão mais sendo utilizadas.

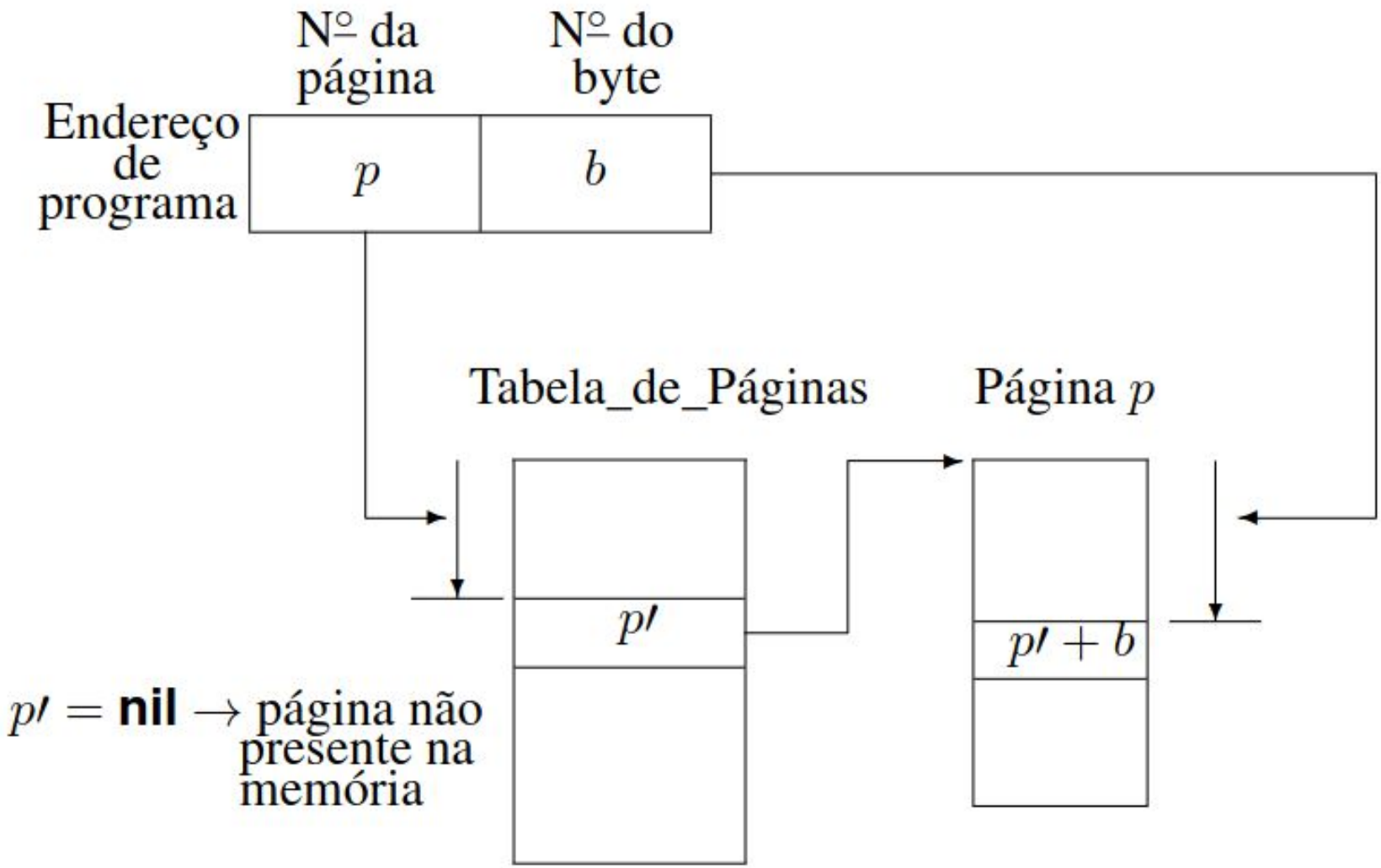
Memória Virtual



Memória Virtual

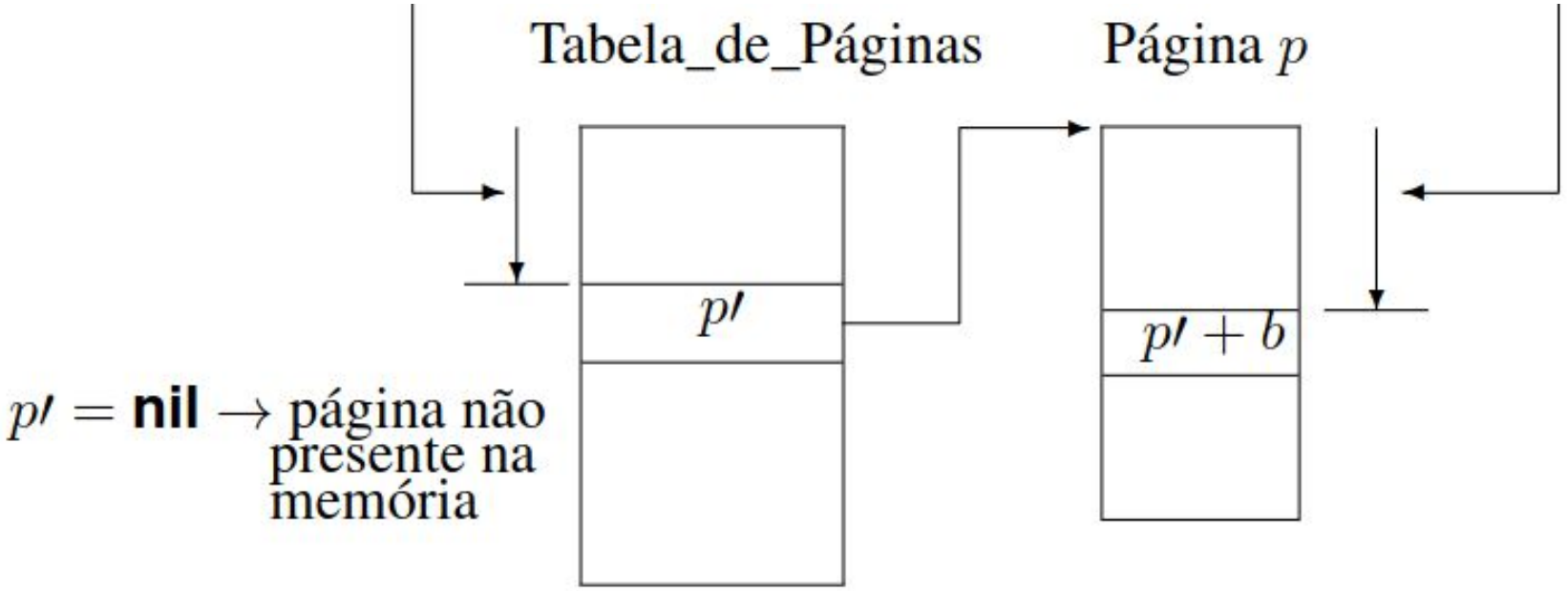
- Endereçamento da página → uma parte dos bits é interpretada como um número de página e a outra parte como o número do byte dentro da página (offset).
- Mapeamento de endereços → realizado através de uma Tabela de Páginas.
  - a p-ésima entrada contém a localização p′ da Moldura de Página contendo a página número p desde que esteja na memória principal.
  - O mapeamento de endereços é:
    $f(e) = f(p, b) = p' + b$,
    onde e é o endereço do programa, p é o número da página e b o número do byte

Memória Secundária



Memória Virtual

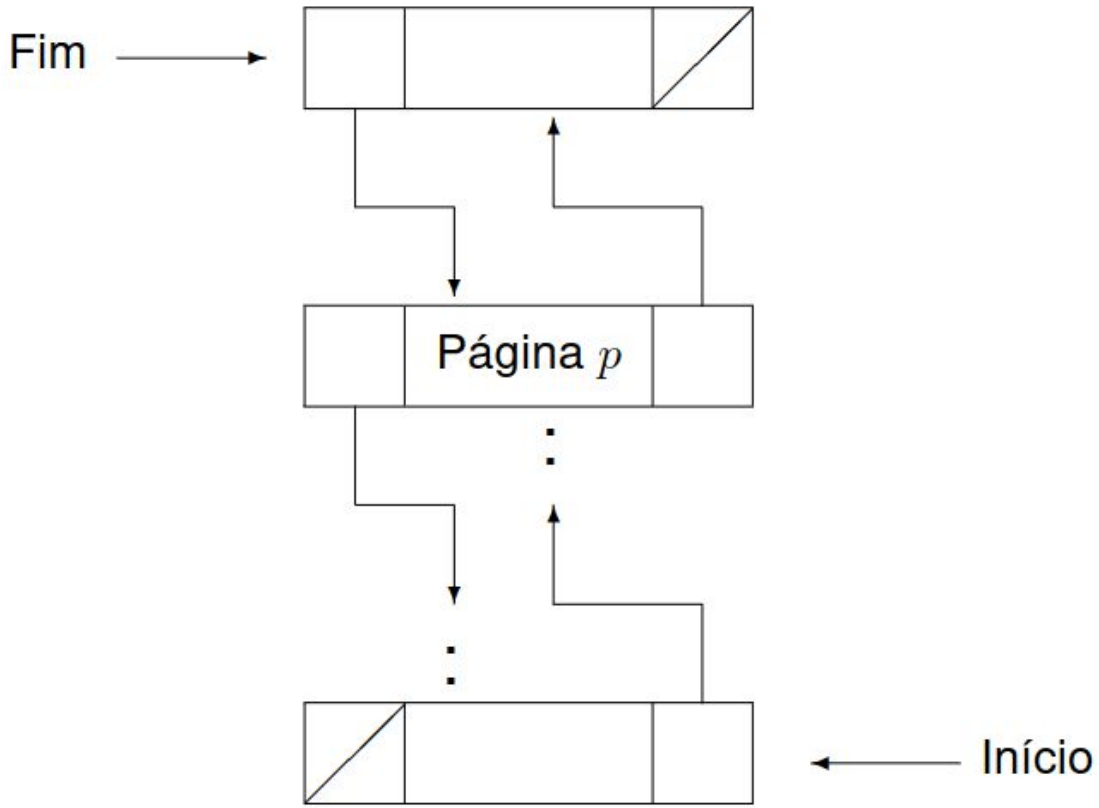
- Se não houver uma moldura de página vazia → uma página deverá ser removida da memória principal.
- Ideal → remover a página que não será referenciada pelo período de tempo mais longo no futuro.
  - tentamos inferir o futuro a partir do comportamento passado



Memória Virtual

- Menos Recentemente Utilizada (LRU):
  - um dos algoritmos mais utilizados,
  - remove a página menos recentemente utilizada,
  - parte do princípio que o comportamento futuro deve seguir o passado recente.
- Menos Freqüentemente Utilizada (LFU):
  - remove a página menos feqüentemente utilizada,
  - inconveniente: uma página recentemente trazida da memória secundária tem um baixo número de acessos e pode ser removida.
- Ordem de Chegada (FIFO):
  - remove a página que está residente há mais tempo,
  - algoritmo mais simples e barato de manter,
  - desvantagem: ignora o fato de que a página mais antiga pode ser a mais referenciada.

Implementação LRU

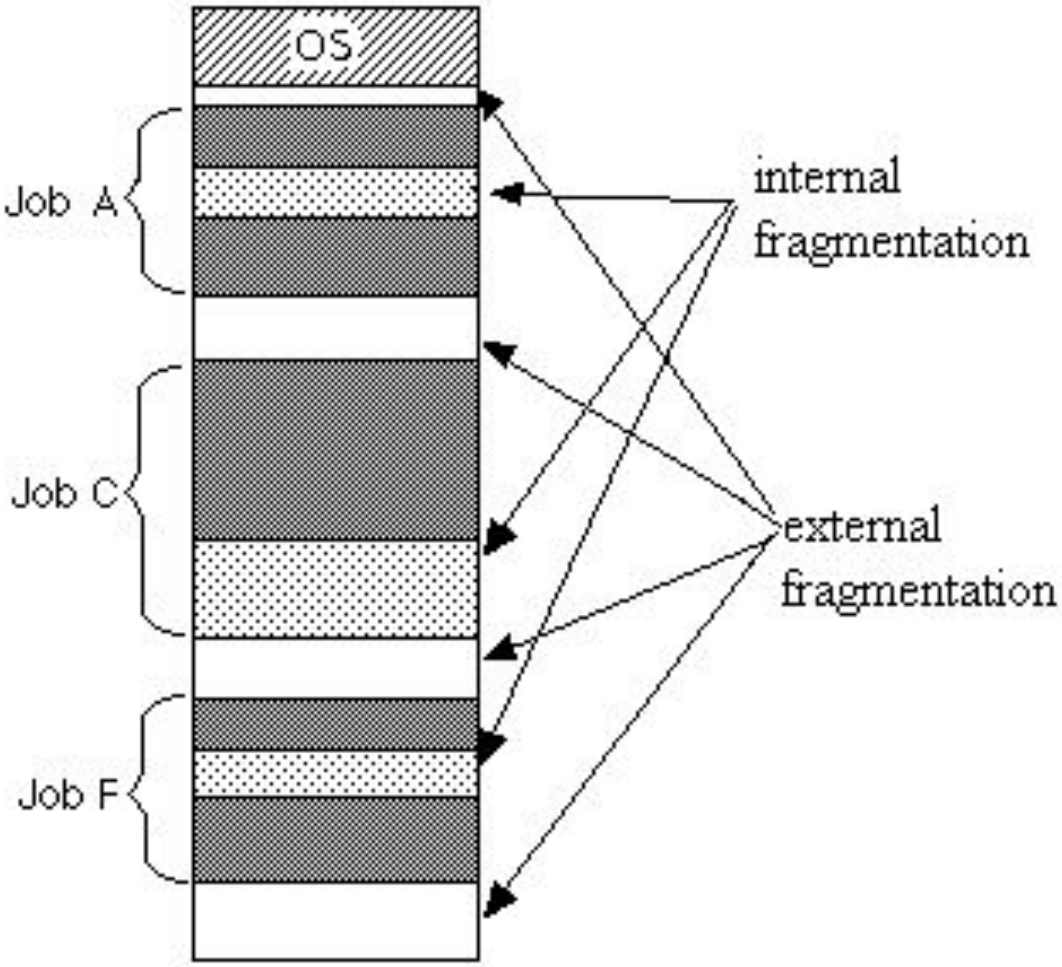


- Toda vez que uma página é utilizada ela é removida para o fim da fila.
- A página que está no início da fila é a página LRU.
- Quando uma nova página é trazida da memória secundária ela é colocada na moldura que contém a página LRU.

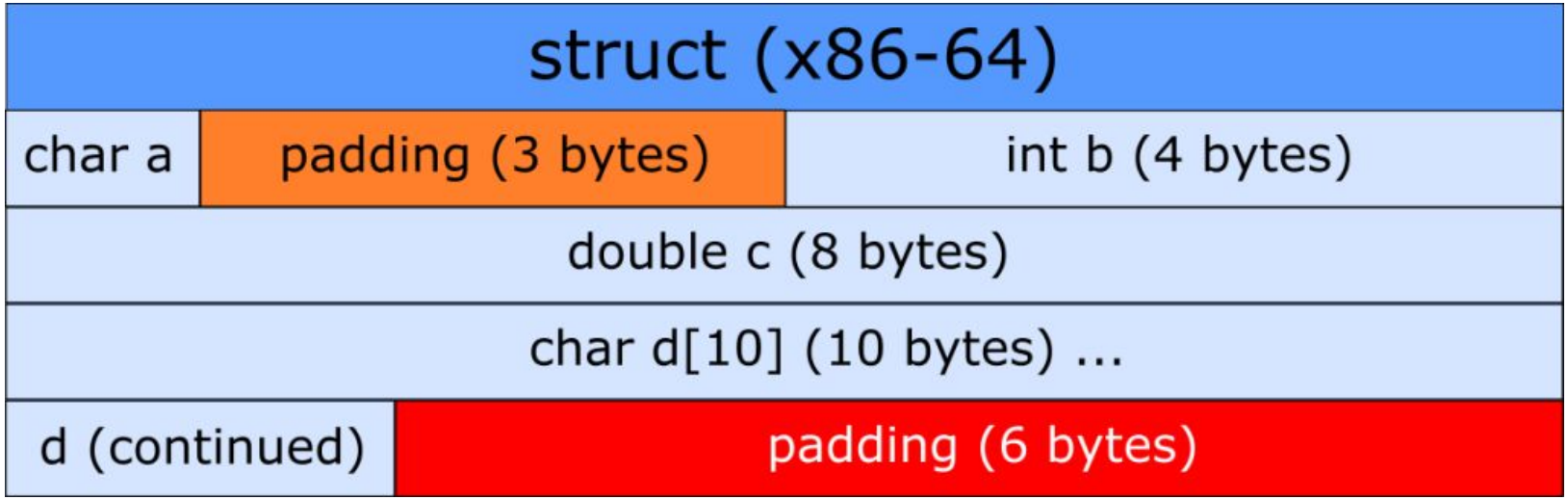
Memória Virtual

```
#define TAMANHODAPAGINA 4096
#define ITENSPORPAGINA 64 // TamanhodaPagina/TamanhodoItem

typedef struct TipoRegistro {
    TipoChave Chave;
    /* outros componentes */
} TipoRegistro ;

typedef struct TipoEndereco {
    long p;
    char b;
} TipoEndereco;

typedef struct TipoItem {
    TipoRegistro Reg;
    TipoEndereco Esq, Dir ;
} TipoItem ;

typedef TipoItem TipoPagina[ ItensPorPagina ] ;
```

Fragmentação de Memória



Alinhamento de Memória



https://abstractexpr.com/2023/06/29/structures-in-c-from-basics-to-memory-alignment/

http://www.catb.org/esr/structure-packing/

Algoritmos para Memória Externa

- Quando se trabalha com memória externa os algoritmos dependem de uma série de fatores, por exemplo:
  - Sistema Operacional
    - Organização da Memória Virtual
    - Mecanismos de Paginação
  - Hardware
    - SSD (Solid State Drives)
    - HDD (Hard Disk Drives)
    - Discos Ópticos
    - Fitas

Algoritmos para Memória Externa

- Vários dos algoritmos são variações dos algoritmos para memória interna

- Ordenação
  - Intercalação Balanceada / Polifásica
  - Seleção por Substituição
  - **Quicksort Externo**

- Pesquisa
  - Acesso Sequencial Indexado
  - **Árvores B**
  - Árvores B*

Acesso Sequencial Indexado

- Utiliza o princípio da pesquisa seqüencial → cada registro é lido seqüencialmente até encontrar uma chave maior ou igual a chave de pesquisa.
- Providências necessárias para aumentar a eficiência:
  - o arquivo deve ser mantido ordenado pelo campo chave do registro,
  - um arquivo de índices contendo pares de valores < x, p > deve ser criado, onde x representa uma chave e p representa o endereço da página na qual o primeiro registro contém a chave x.

Acesso Sequencial Indexado

Estrutura de um arquivo seqüencial indexado para um conjunto de 15 registros:

3	14	25	41
1	2	3	4

1 | 3 5 7 11 | 2 | 14 17 20 21 | 3 | 25 29 32 36 | 4 | 41 44 48 |

Pesquisa em Memória Secundária

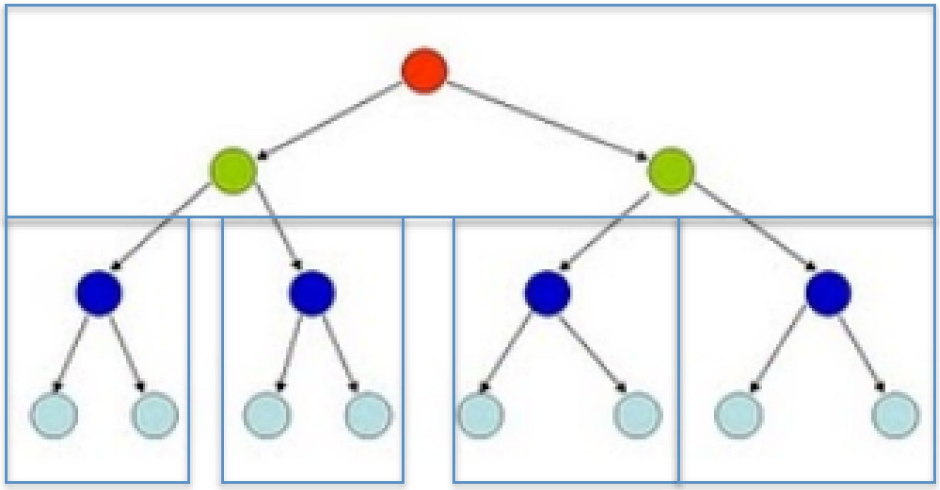
- Arquivos contém mais registros do que a memória interna (primária) pode armazenar
- Custo para acessar um registro é ordens de grandeza maior que o custo de acesso à memória primária
- **Medida de complexidade**: custo de transferir dados entre a memória principal e secundária (minimizar o número de transferências)

Pesquisa em Memória Externa

- **Problema**: acessar dados em arquivos grandes armazenados em memória secundária
- **Solução 1**: Usar uma árvore binária
  - Armazenar nós em disco
  - Ponteiros *esq* e *dir* apontam para endereços em disco
  - Custo de leitura: O(log n) acessos em disco
    - n = 10^6, log(n) ≈ 20 acessos em disco!

Pesquisa em Memória Externa

- Agrupar nós da árvore binária em páginas



- Problema: qual a melhor forma de distribuir os registros entre as páginas? (problema de otimização complexo)

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Estruturas de Dados

Memória Secundária

Professores: Anisio Lacerda
Lucas Ferreira
Wagner Meira Jr.
Washington Cunha