# GBC053 - Gerenciamento de Bancos de Dados

Aula 3
Armazenamento Secundário e Sistemas de Softwares

Humberto Razente
humberto.razente@ufu.br

# Discos

- Comparado com o tempo que leva para acessar um item na RAM, acessos a disco são sempre caros
  - entretanto, nem todos os discos são igualmente lentos
- Discos rígidos:
  - Direct Access Storage Devices
  - permitem acessar dados diretamente
    - em oposição aos dispositivos de fita nos quais é preciso percorrer a fita até alcançar uma posição

# Mídias magnéticas



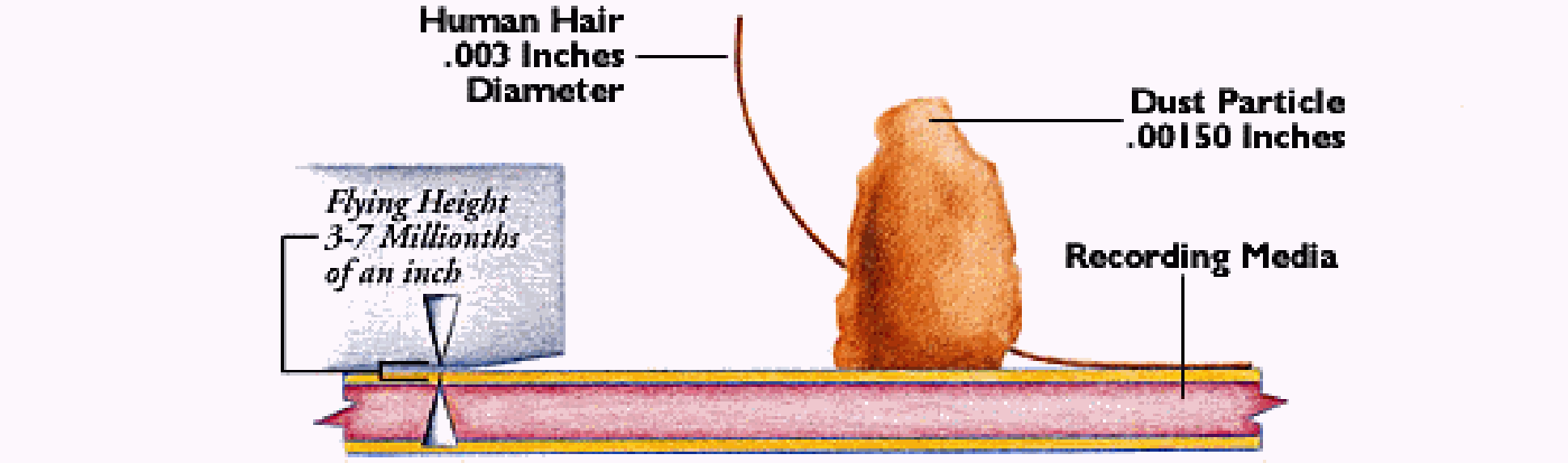


# Mídias óticas

# Discos rígidos

Discos de 8″, 5.25″, 3.5″, 2.5″, 1.8″ e 1″

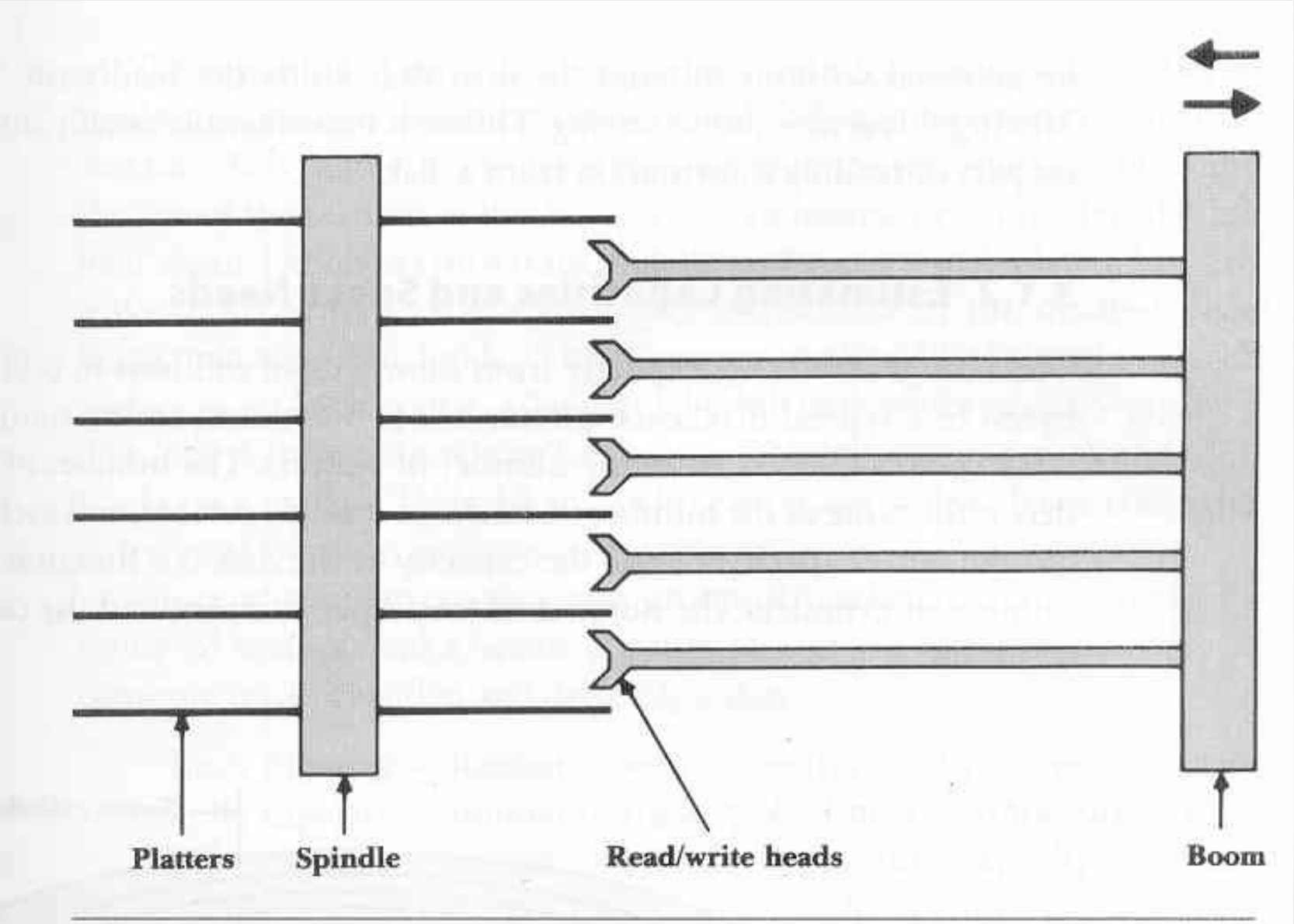
# Discos rígidos



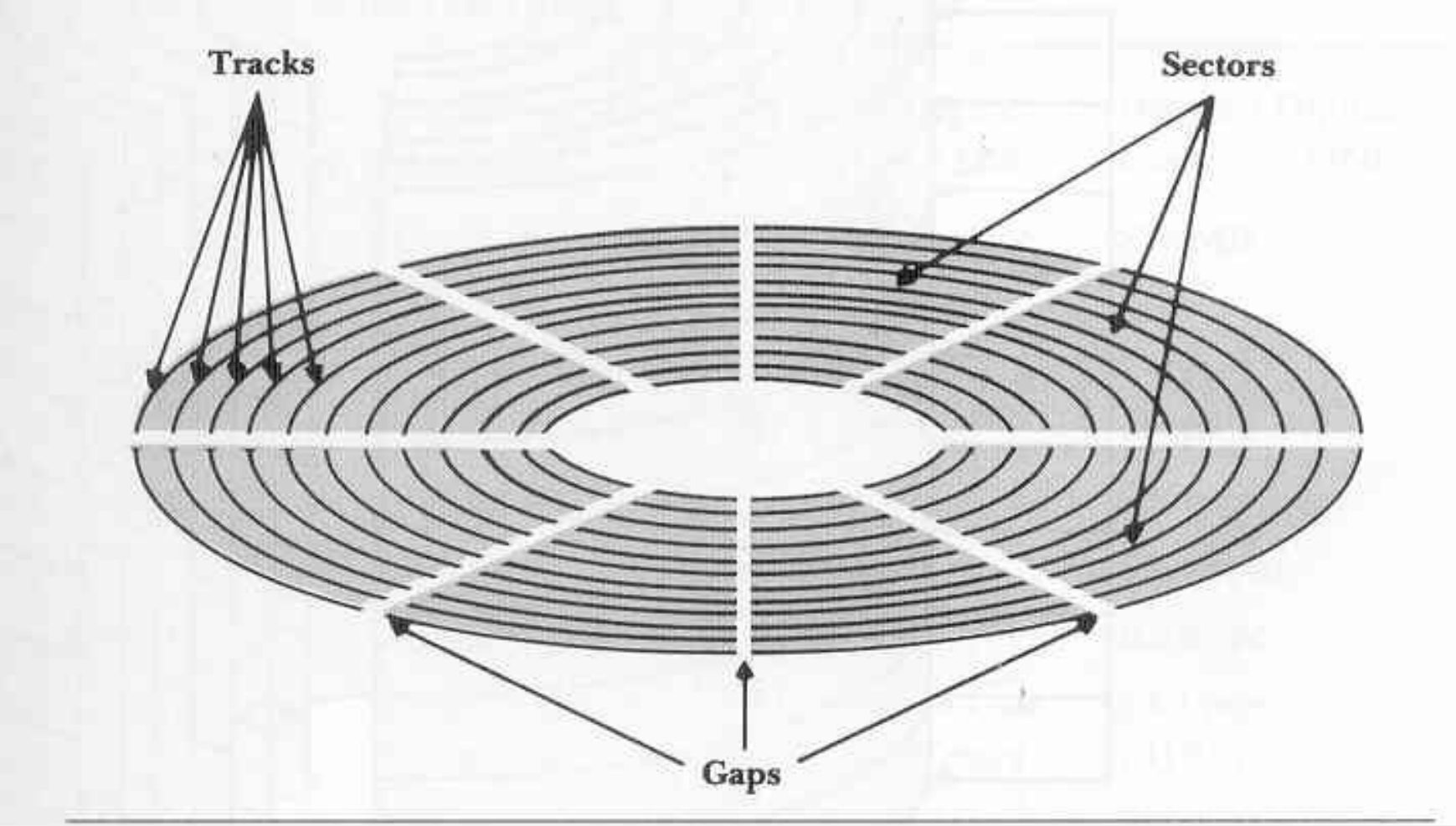
# Organização de discos

- Disco rígido
  - superfície tem um ou mais pratos
  - arquivos são armazenados em trilhas (*tracks)* sucessivas
  - Trilhas são divididas em setores (sectors)
  - Um setor é a menor porção endereçável em um disco
    - quando um READ tenta ler 1 byte de um arquivo, o sistema operacional encontra o prato, trilha e setor, lê o setor inteiro para uma área da RAM chamada BUFFER, e então recupera o byte e retorna à aplicação

# Disco rígido



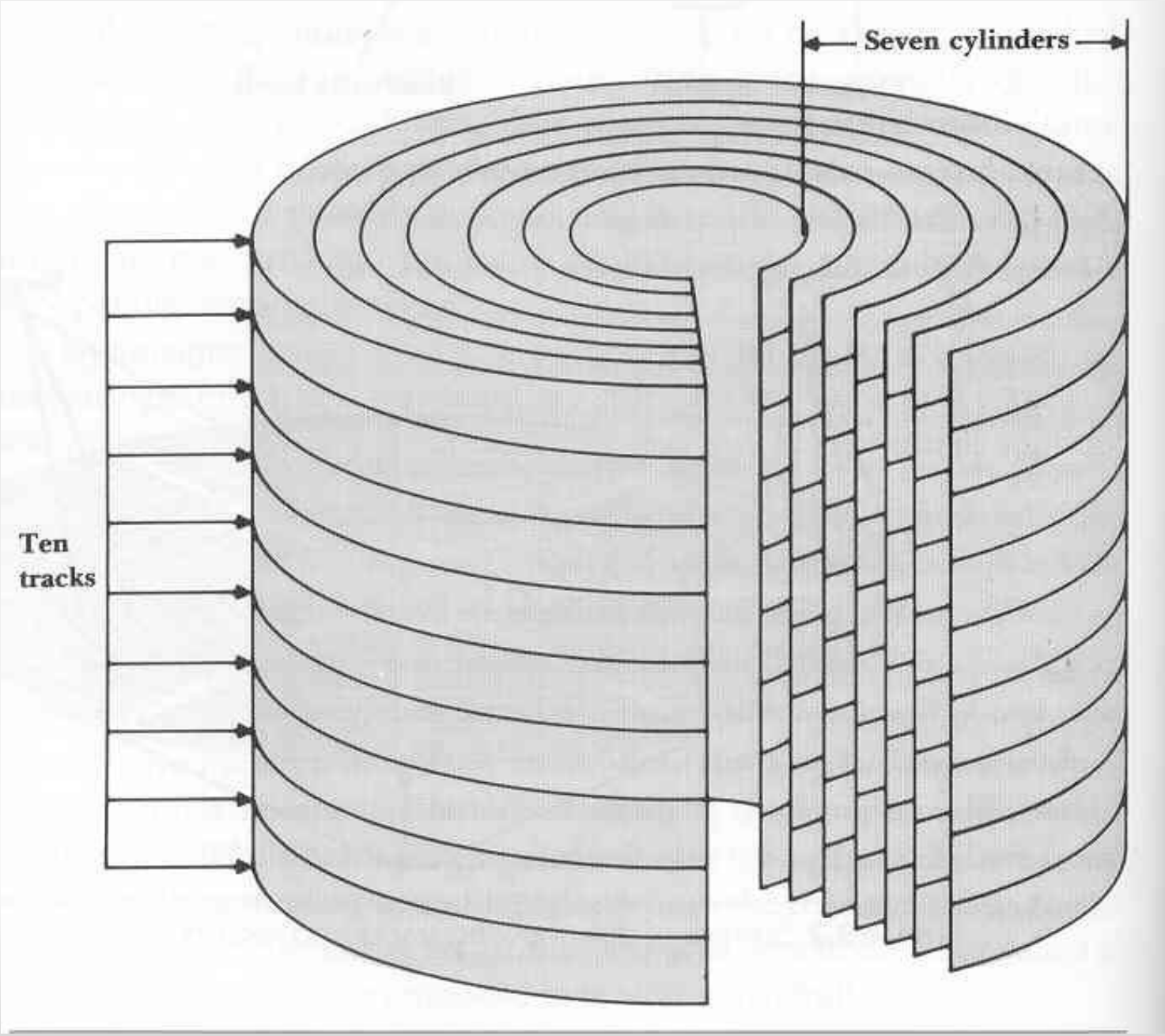
# Trilhas e setores



# Cilindros

- Cilindro: trilhas diretamente acima ou abaixo formam um cilindro
  - importância: informações em um cilindro podem ser acessadas sem a movimentação do braço que segura as cabeças de leitura e gravação
  - movimentação do braço é chamado *seeking*
    - usualmente a tarefa mais lenta da leitura

# Cilindros



# Capacidades

- trilha = número de setores por trilha x bytes por setor
- cilindro = número de trilhas por cilindro x capacidade da trilha
- drive = número de cilindros x capacidade do cilindro

# Exemplo

- Armazenar um arquivo de 20.000 registros de tamanho fixo de um disco rígido de 300 MB com as seguintes características
  - bytes por setor = 512
  - setores por trilha = 40
  - trilhas por cilindro = 11
  - cilindros = 1.331
- Quantos cilindros são necessários para armazenar o arquivo se um registro tiver 256 bytes?

# Exemplo

- 20.000 registros x 256 bytes = 10.000 setores
  512 bytes por setor

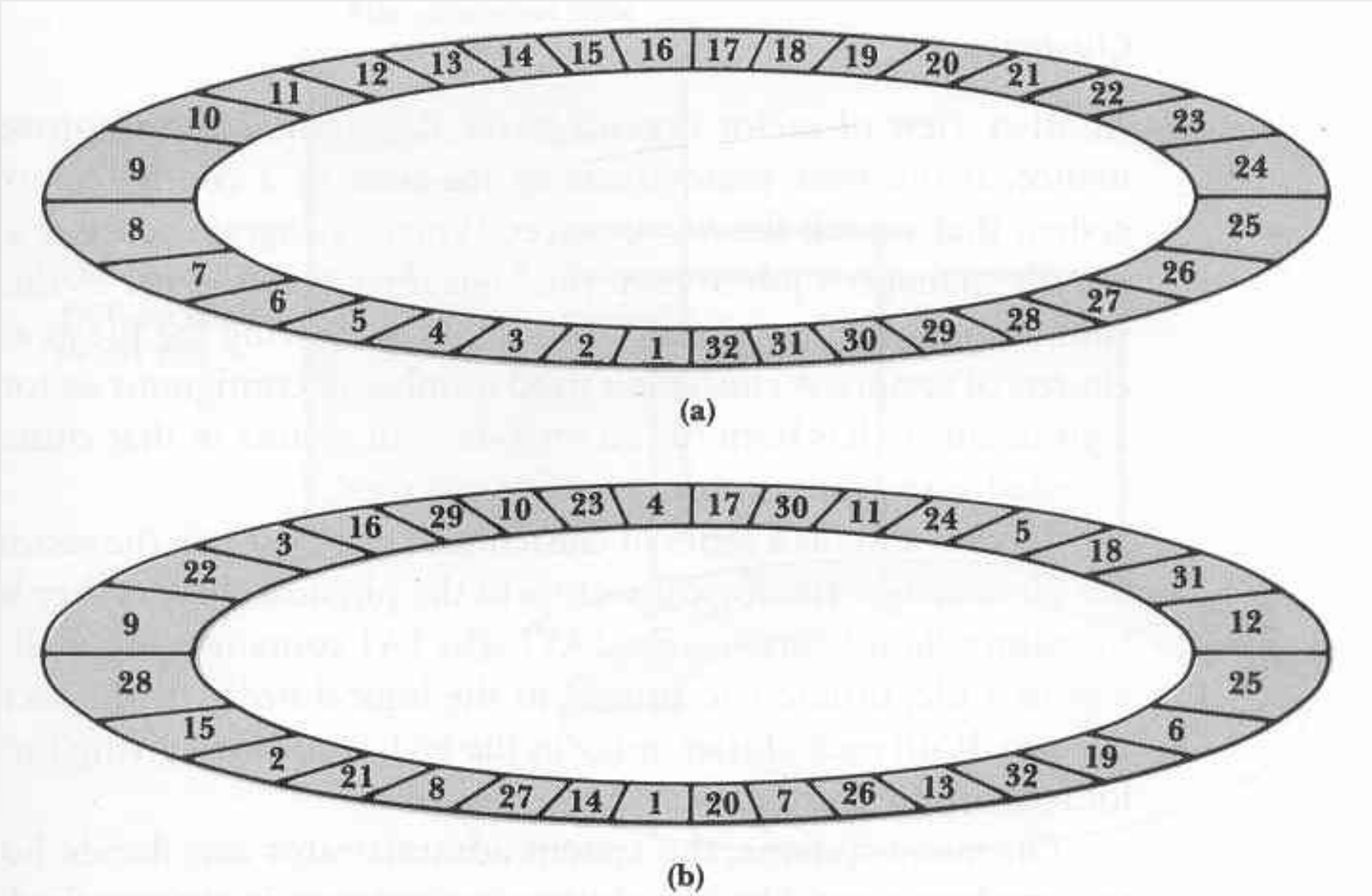
- 1 cilindro = 40 x 11 = 440 setores

- Logo, 10.000 / 440 = 22,7 cilindros

# Blocos e setores

- Há basicamente 2 modos de organizar dados em um disco:
  - por setor
    - setores são segmentos adjacentes de uma trilha
  - por bloco definido pelo usuário
    - ainda na década de 90, muitas controladoras de disco não tinham capacidade suficiente para ler um setor, guardá-lo em BUFFER, e ler outro setor sem ter que esperar o disco executar uma rotação
    - opção era organizar blocos em setores não adjacentes → tarefa do sistema de arquivos, de modo transparente para o programador

# Blocos e setores




- (b) fator *interleaving* = 5 → ao invés de ler os 32 setores com 32 rotações do disco, 5 rotações são necessárias

# Fragmentação

- Em geral, todos os setores de um disco têm o mesmo número de bytes
- Se, por exemplo, tivermos um setor de 512 bytes e registros de 300 bytes
  - podemos guardar um registro por setor
    - resultado: fragmentação interna
  - permitir que registros ultrapassem o setor, continuando no próximo
    - início de um registro pode ser encontrado em um setor e final em outro
    - adequado quando acesso é sequencial

# Custo do acesso a disco

- Três operações físicas distintas:
  - tempo de seek
  - atraso rotacional
  - tempo de transferência

# Tempo de seek

- Tempo de seek
  - movimentação do braço da cabeça de leitura/escrita para o cilindro correto
  - depende do número de cilindros a deslocar
- Se um arquivo estiver armazenado em blocos consecutivos e em cilindros consecutivos
  - leitura ideal, com tempo de seek de 1 trilha a cada leitura de vários setores
- Acesso simultâneo a dois arquivos armazenados em cilindros nos extremos
  - pior caso

# Tempo de seek

- Muito custoso em ambientes multi-usuário
  - muitos processos simultâneos com uso de disco
  - alto custo para comportamento aleatório
- Como é usualmente impraticável determinar o tempo de seek para cada operação, em geral computa-se o ”tempo médio de seek” para uma operação de arquivo em particular
  - Em 1991: 40 ms
  - Em 2009: até 2 ms, desktop comum 9 ms, notebook 15 ms
  - Muito grande em mídias óticas

# Atraso rotacional

- Atraso rotacional
  - tempo necessário para o disco rotacionar até que o setor desejado esteja embaixo da cabeça de leitura/gravação
  - atualmente, a grande maioria dos HD para desktops tem velocidade de 7.200 rotações por minuto (IDE ou SATA)
    - 5.400 é padrão em notebooks
    - 10.000 e 15.000 rpm disponíveis em HDs de alta performance (padrão SAS ou SCSI)

# Atraso rotacional

- Atraso rotacional
  - em média, atraso rotacional é de meia volta
  - alto custo para comportamento aleatório

# Tempo de transferência

- Tempo de transferência
  - tempo que leva a leitura de um setor que está sob a cabeça de leitura/gravação

    $$\frac{\text{número de bytes transferidos}}{\text{número de bytes na trilha}} \times \text{tempo rotação}$$

  - Para calcular por setor, divide-se pelo número de setores por trilha

# Custo do acesso a disco

- Tempo médio para leitura de um arquivo =

  tempo de seek

  +

  atraso rotacional

  +

  tempo de transferência

# O disco visto como um gargalo

- Mesmo com o desempenho dos discos crescendo rapidamente, tal velocidade ainda perde para as velocidades das redes locais
  - resultado: CPU e rede têm que esperar grande quantidade de tempo para disco atender às requisições
- Uma alternativa é a técnica de *stripping*
  - envolve quebrar um arquivo em partes e colocá-las em discos diferentes (de modo transparente para o usuário)

# O disco visto como um gargalo

- Com o custo das memórias RAM caindo, outra alternativa  é criar discos virtuais em memória
  - um RAM Disk é uma grande parte da memória RAM configurada para simular um disco mecânico em todos os aspectos exceto
    - velocidade e volatilidade

# O disco visto como um gargalo

- Cache
  - o cache de disco é um grande bloco de memória RAM configurado para conter páginas ou setores de um disco
    - quando dados são requisitados, primeiro verifica-se se está no cache, senão busca-o no disco, incluindo ou substituindo no cache

# Fita magnética

- Não provê a facilidade do acesso direto aos dados

- Mas pode prover acesso sequencial rápido
  - fitas são compactas, fáceis de transportar
  - são muito menos custosas que discos

# Fita magnética – aplicações

- Mídia apropriada para processamento sequencial se
  - arquivos processados não serão utilizados por aplicações que requerem acesso direto
  - exemplo: contribuintes tem prazo para enviar declarações de imposto de renda
    - processamento pode ser feito após encerramento do prazo
  - com o constante barateamento dos discos rígidos por MB armazenado, atualmente é raro o processamento de dados diretamente em fita

# Disco rígido *vs* fita magnética

- No passado, discos e fitas dividiam o mercado de mídias para armazenamento secundário
  - disco permitia acesso aleatório e imediato
  - fita permitia processamento sequencial e armazenamento de longo prazo
- Com o tempo, a relação tendeu em favor dos discos
  - disco permite acesso multi-usuário
- Ainda assim, as fitas são utilizadas para armazenamento de longo prazo (backup)

# Hierarquia de armazenamento

| Tipos de memória | Dispositivos e mídias |
| --- | --- |
| **Primárias**<br>Registradores<br>RAM<br>RAM disk<br>Disk cache | Núcleos (cores) e semicondutores |
| **Secundárias**<br>Acesso direto<br>Sequencial | Discos magnéticos<br>Fitas |
| **Offline**<br>Backup | Discos magnéticos removíveis, mídias óticas, fitas |

# A jornada de um byte

- O que acontece quando um programa escreve um byte em um arquivo em disco?
  - WRITE(...);
- O comando WRITE resulta em uma chamada ao sistema operacional
  - que tem a função de acompanhar o restante da jornada até que seja completada com sucesso

# A jornada de um byte

- Gerenciador de arquivos: componente do SO
  - várias camadas de procedimentos
  - camadas superiores lidam com aspectos lógicos
  - camadas inferiores lidam com aspectos físicos
    - cada camada fala com a inferior, até que a última camada realmente escreve no disco
- Funcionamento
  - determinar se arquivo está aberto, se usuário tem permissão, etc
  - determinar local onde o byte será colocado
    - localização física do setor onde será gravado

# A jornada de um byte

- A seguir, o gerenciador de arquivos determina se o setor está na RAM ou precisa ser lido
  - denominado I/O buffer
- Então o byte pode ser incluído na posição
- O sistema de I/O buffer controla se o setor foi gravado ou não de volta no disco
  - idéia é adiar a gravação, permitindo que outros bytes possam ser gravados sem acesso ao disco
  - usuário pode requisitar *flush* para persistir buffer ou fechar arquivo
    - ambos resultam na gravação do setor no disco

# Gerenciamento de buffer

- Buffering envolve trabalhar com grandes pedaços de dados em RAM para que o número de acessos ao armazenamento secundário seja reduzido

- Suponha que há apenas 1 buffer disponível:
  - uma aplicação que lê 1 byte por vez de um arquivo e grava em outro arquivo
    - geraria um grande número de leituras/gravações (um par para cada byte)
    - torna-se óbvio que o gerenciador de buffers deva poder gerenciar várias páginas simultaneamente

# Gerenciamento de buffer

- Estratégias
  - buffering múltiplo (multiple buffering)
  - move mode and locate mode
  - scatter/gather I/O

# Leitura complementar

- Capítulo ”Secondary storage and system storage” do livro
  - Folk et al. ”File Structures: An Object-Oriented Approach with C++”, Editora Pearson, 3ª edição, 1998