ACH2024 –

Algoritmos e Estruturas de Dados II

Prof. Helton Hideraldo Biscaro
heltonhb@usp.br

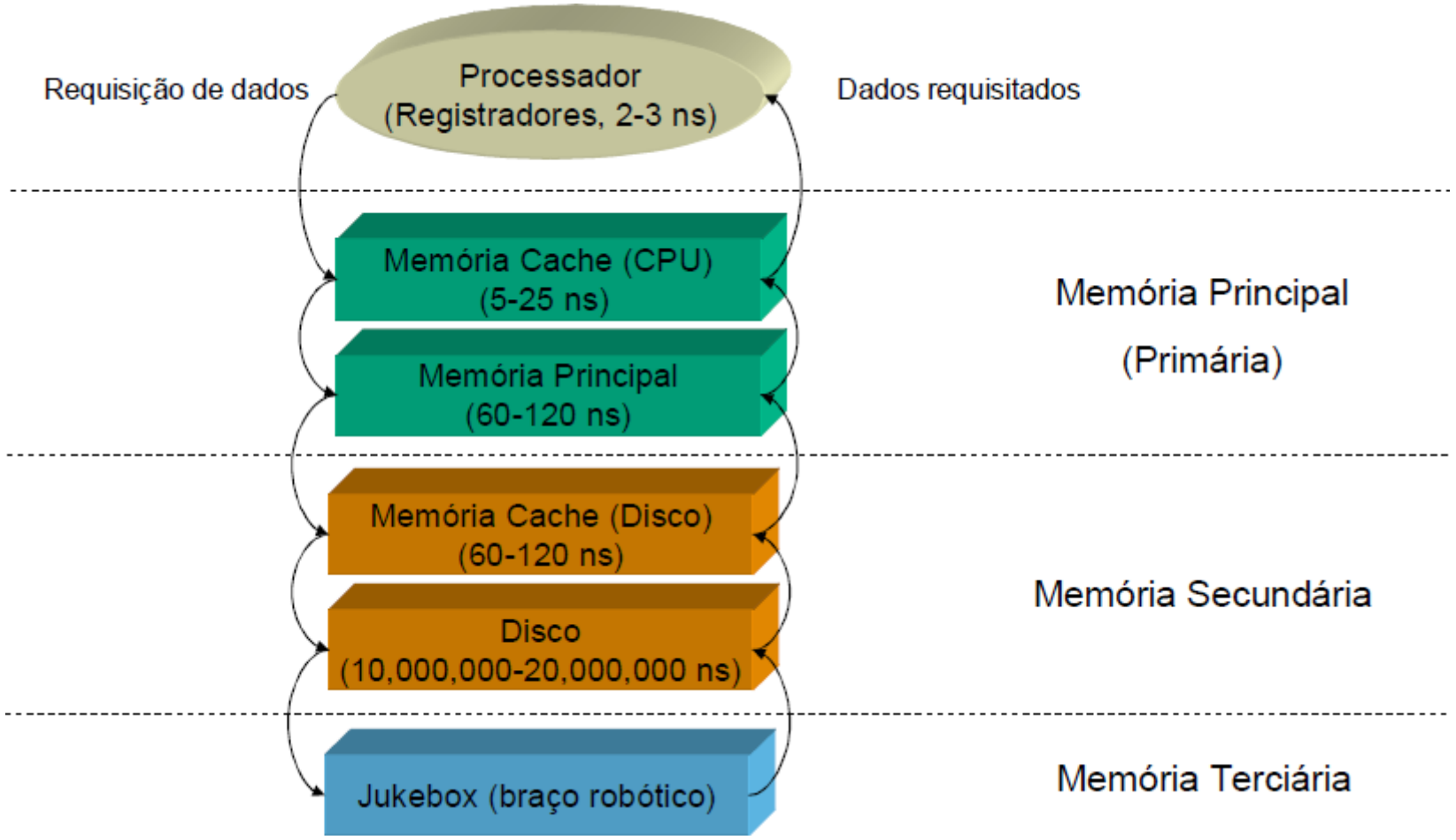
Parte 2 da Disciplina:

Memória Secundária

Organização de Arquivos

Nesta aula são introduzidos conceitos sobre discos e organização de arquivos e o modelo consequencial de processamento de arquivos

Níveis de Armazenamento



Memória Cache

A memória **cache** corresponde a uma memória de acesso mais rápido do que a memória usual

Por ser mais rápida seu custo é mais elevado

- O tamanho da memória **cache** geralmente é bem menor que a memória usual

Na memória **cache** é colocada uma (pequena) cópia dos dados originais (armazenados na memória usual ou computados anteriormente)

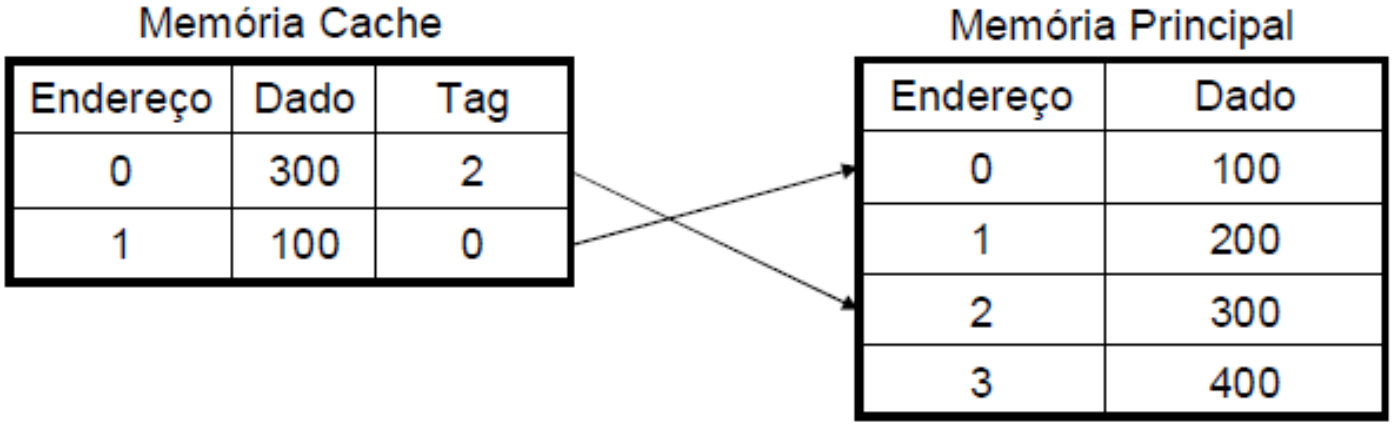
- Somente os dados mais frequentemente utilizados permanecem no cache

Caches mostram-se extremamente eficientes em muitas áreas da computação devido ao fato que os padrões de acesso em aplicações típicas possuem uma certa localização das referências

Memória Cache

O funcionamento básico de memórias **cache** pode ser resumido da seguinte forma:

- Se o dado solicitado encontra-se no **cache**, utilize-o (*cache hit*)
- Se o dado solicitado não se encontra no **cache**, traga-o para o **cache** e utilize-o (*cache miss*)



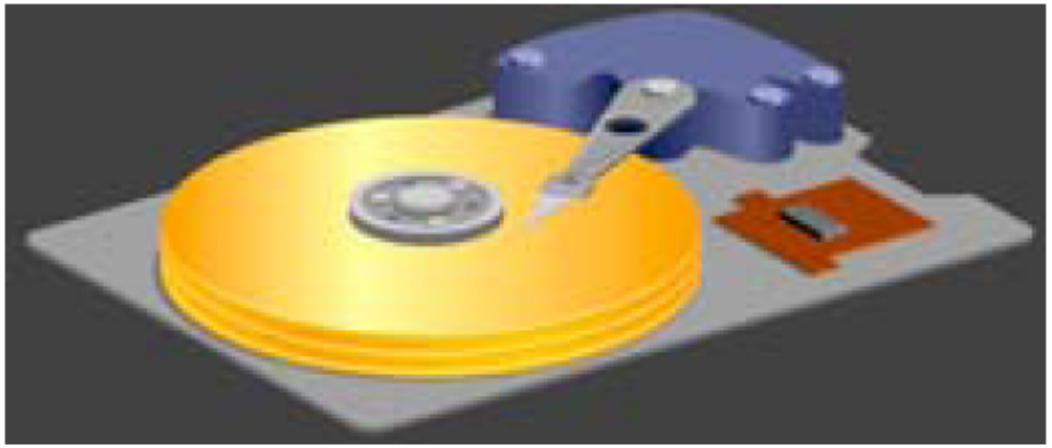
Prefixos Binários

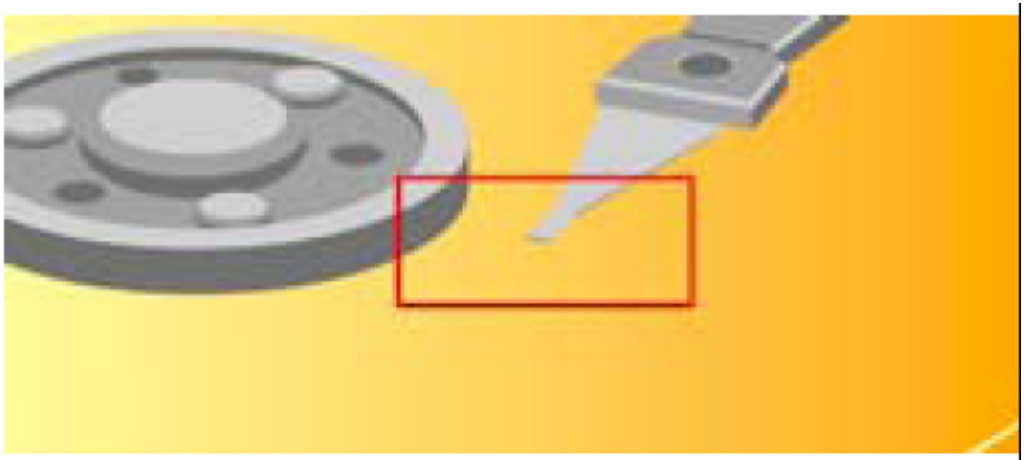
Prefixo	Símbolo	Valor	Base 10
kilo	k/K	2^{10} = 1,024	> 10^{3}
mega	M	2^{20} = 1,048,576	> 10^{6}
giga	G	2^{30} = 1,073,741,824	> 10^{9}
tera	T	2^{40} = 1,099,511,627,776	> 10^{12}
peta	P	2^{50} = 1,125,899,906,842,624	> 10^{15}
exa	E	2^{60} = 1,152,921,504,606,846,976	> 10^{18}
zeta	Z	2^{70} = 1,180,591,620,717,411,303,424	> 10^{21}
yota	Y	2^{80} = 1,208,925,819,614,629,174,706,176	> 10^{24}

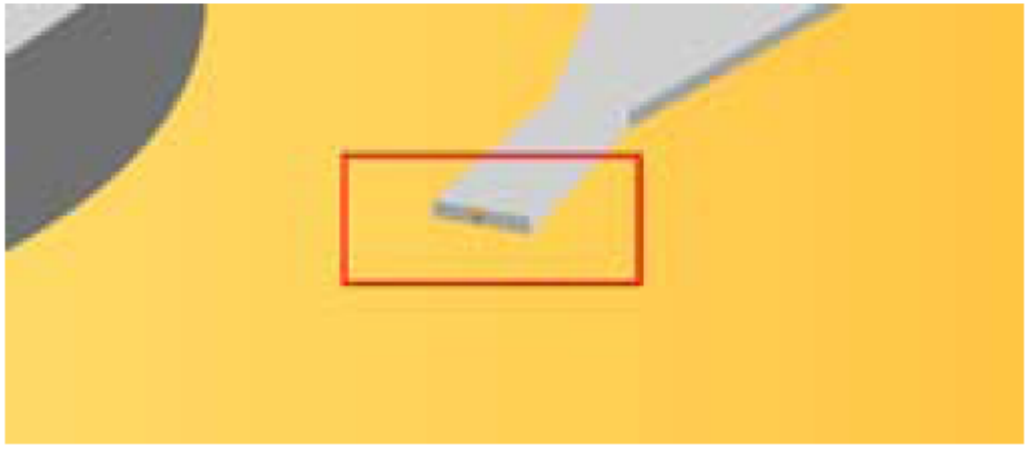
Discos

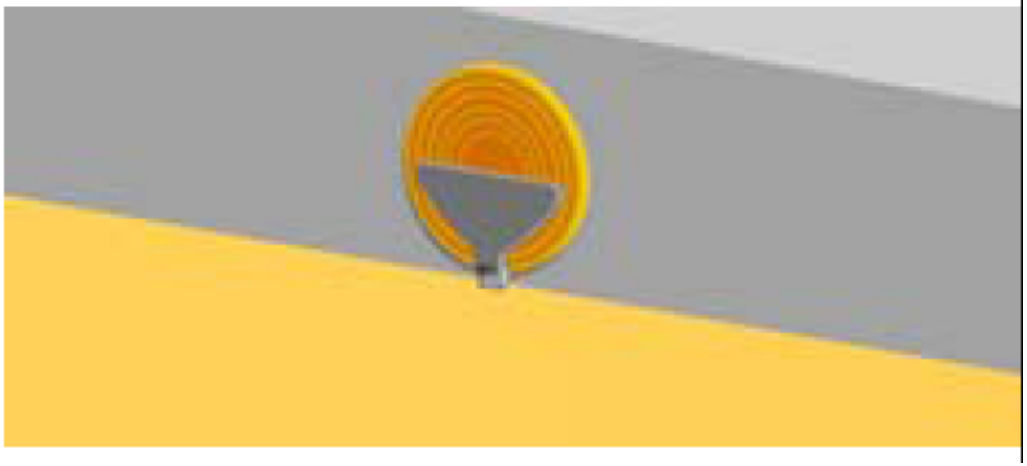
Vamos nos concentrar principalmente nos dispositivos de memória secundária de acesso direto, exemplificados pelos discos

Discos

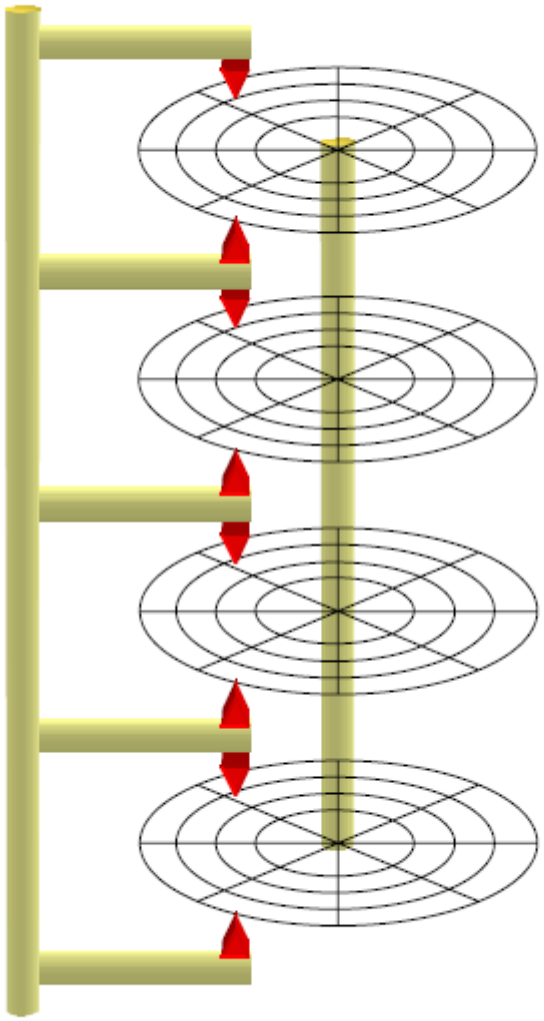






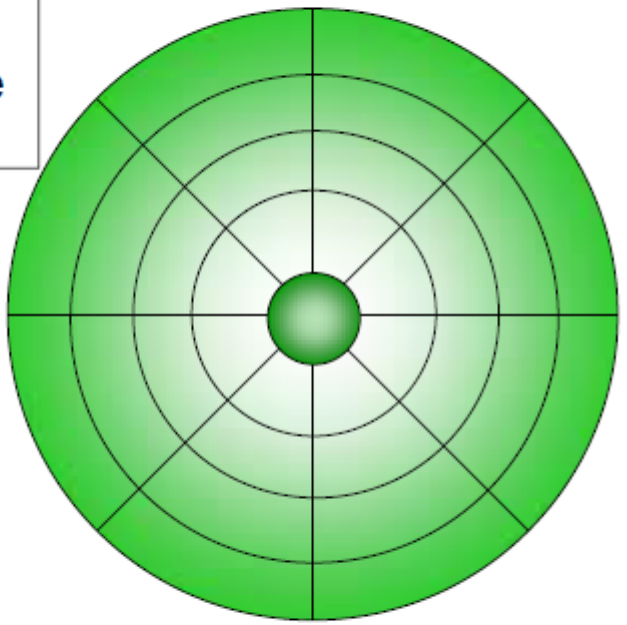


Terminologia sobre Discos

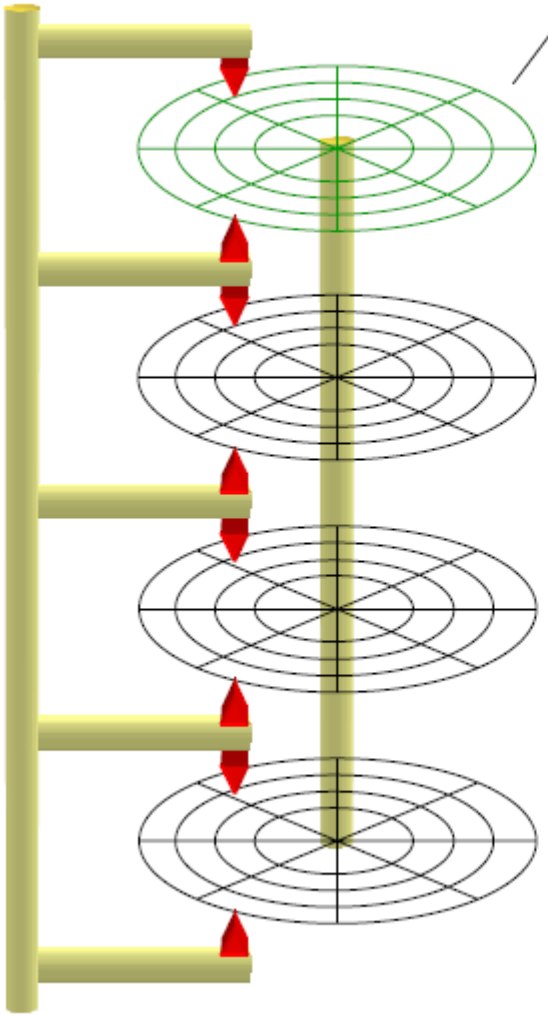


Braço

Dispositivo que suporta as cabeças de leitura/gravação à medida que elas se movem na superfície do disco

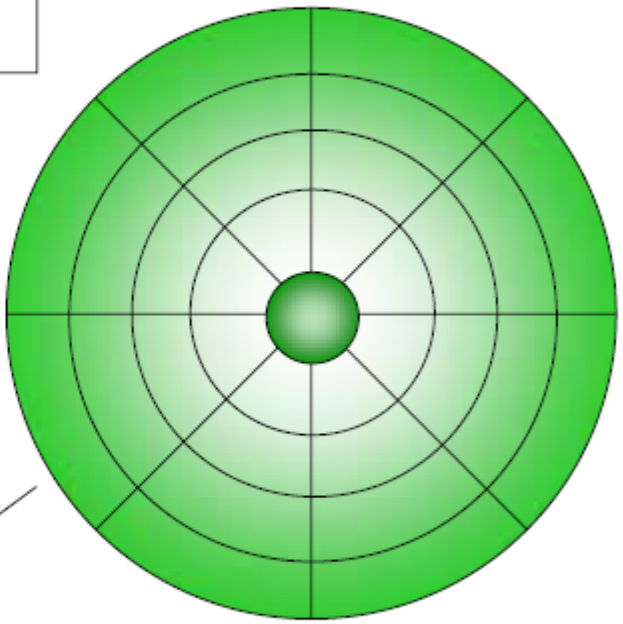


Terminologia sobre Discos



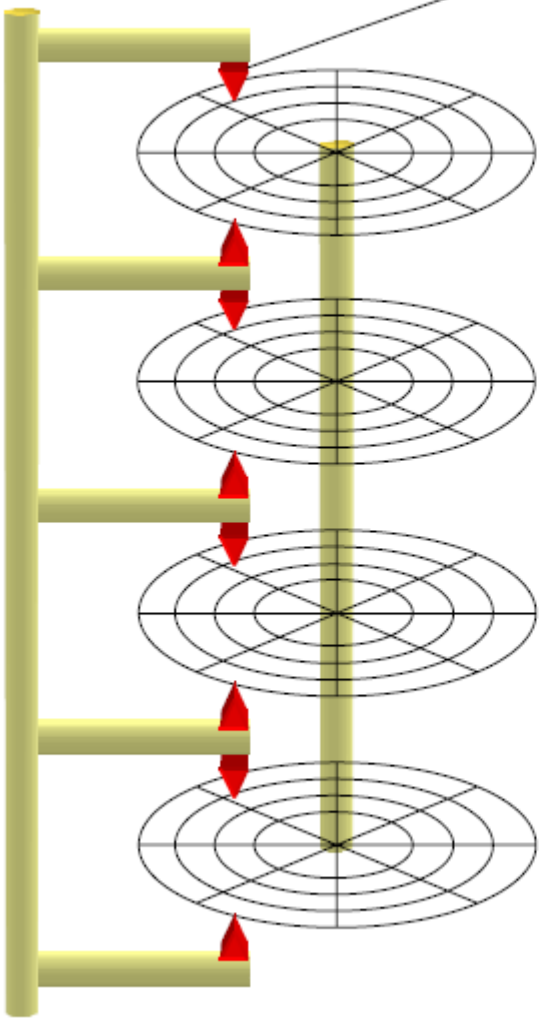
Prato (disco individual)

Um disk drive pode ter vários pratos (discos)



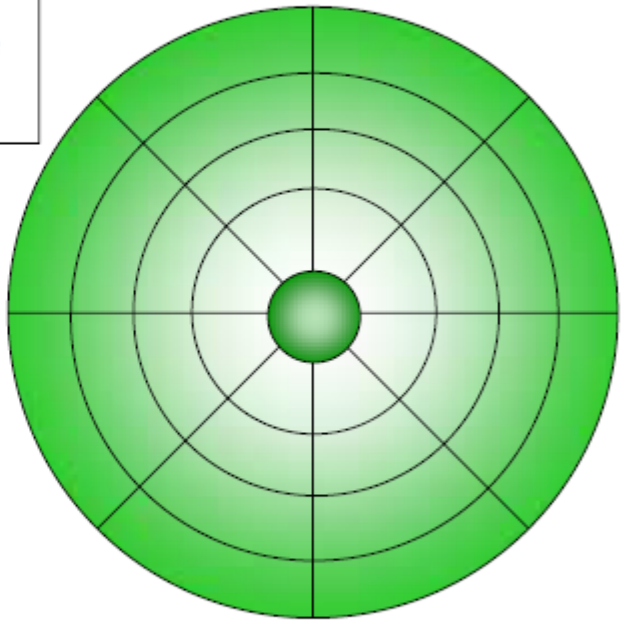
Prato

Terminologia sobre Discos

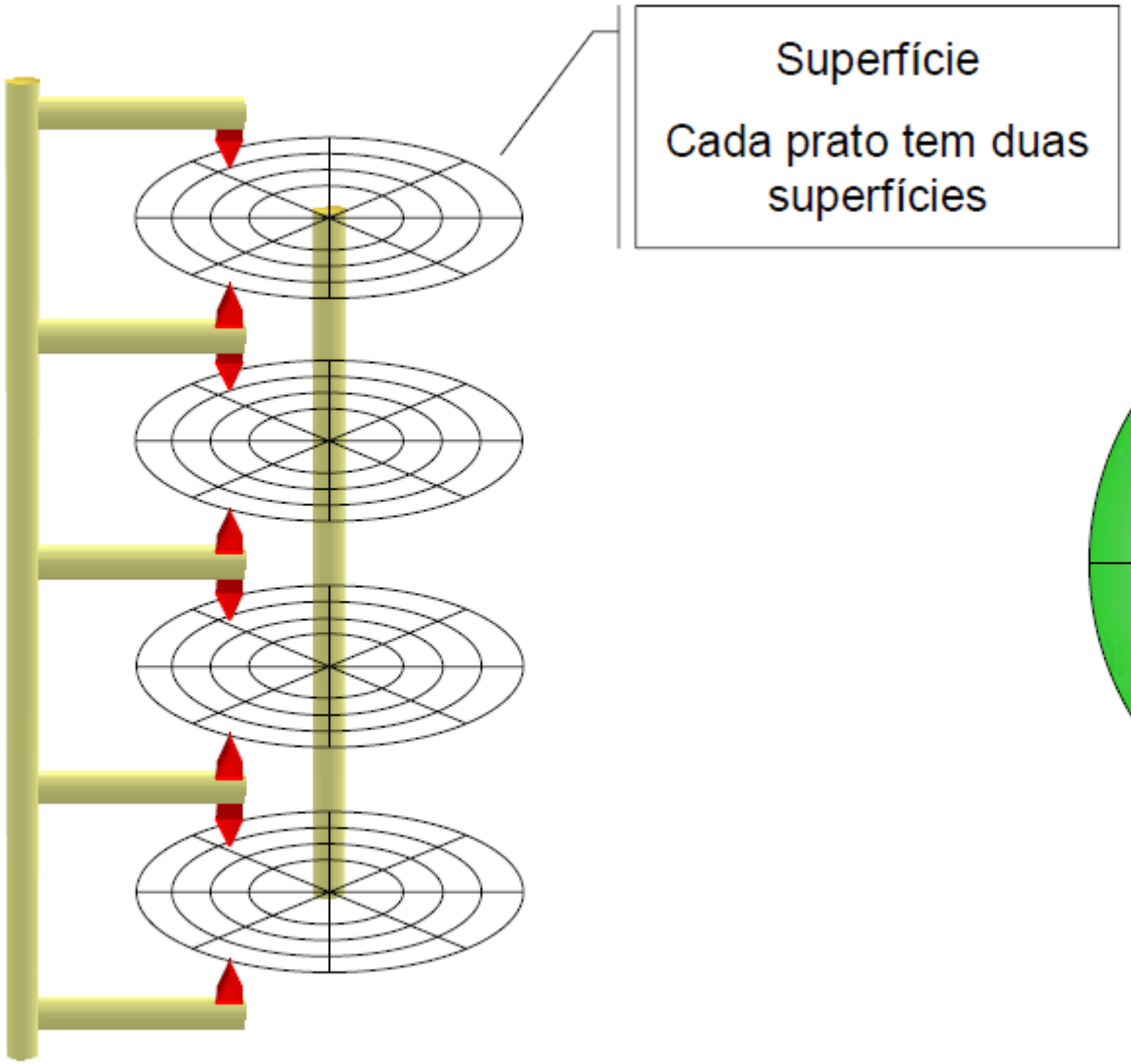


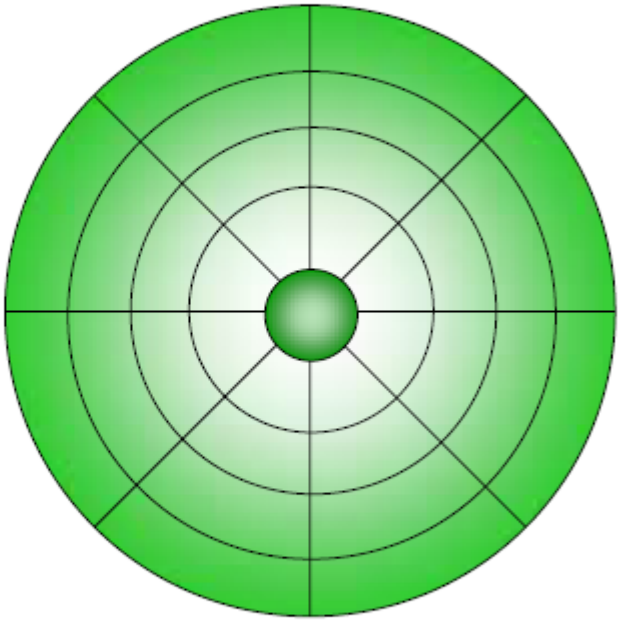
Cabeça de leitura/gravação

Dispositivo que realiza a leitura e escrita na superfície de armazenamento

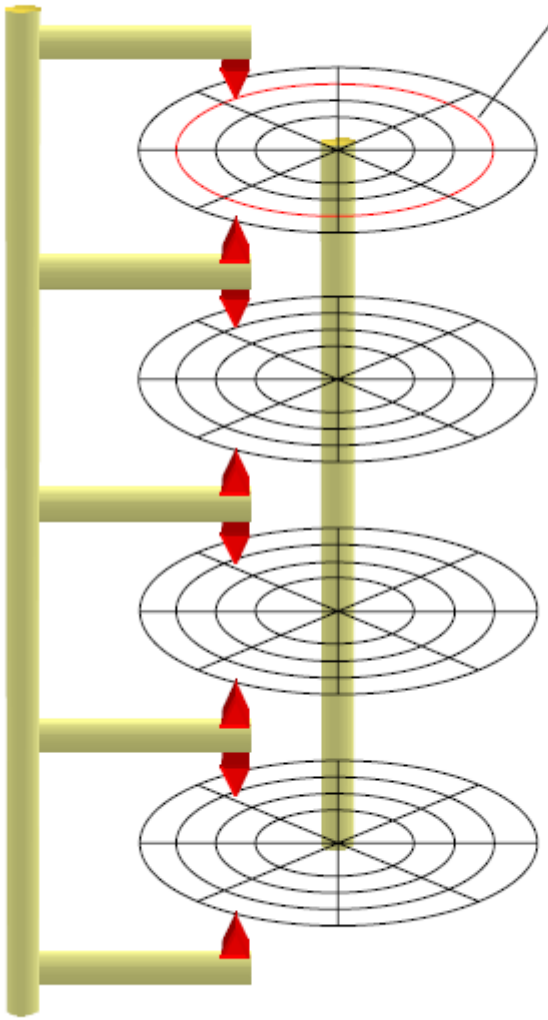


Terminologia sobre Discos



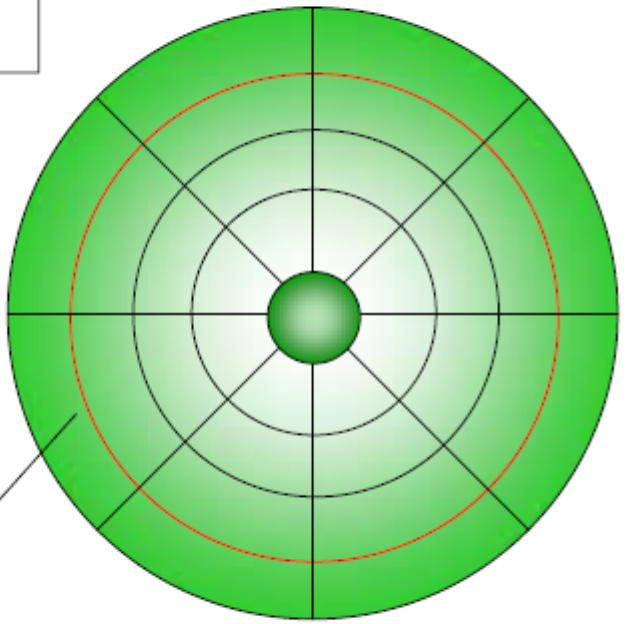


Terminologia sobre Discos



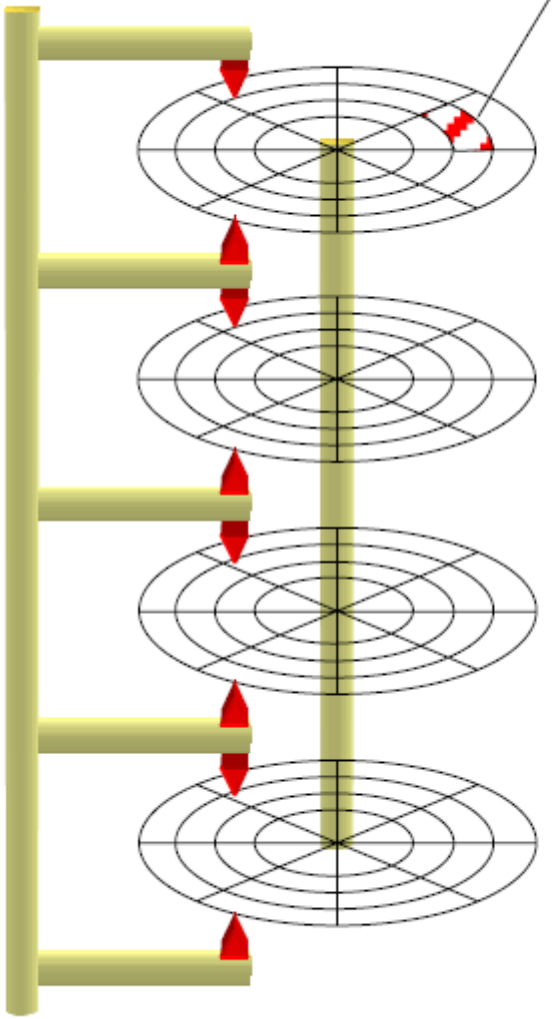
Trilha

Um círculo em uma única superfície de armazenamento



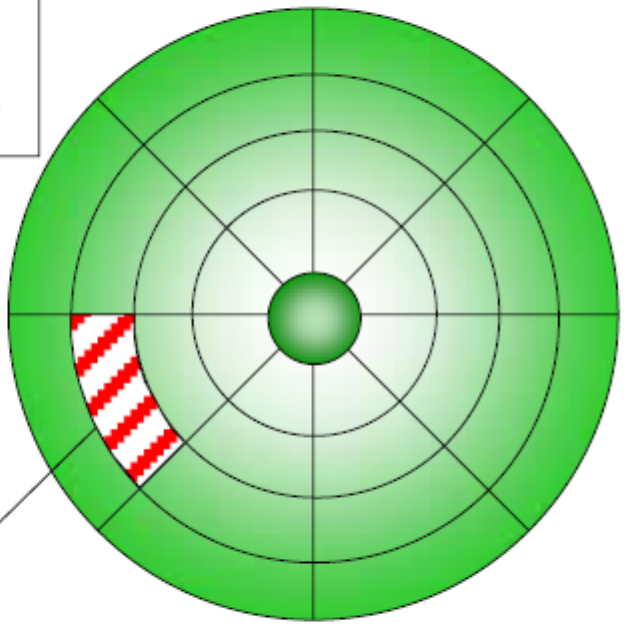
Trilha

Terminologia sobre Discos



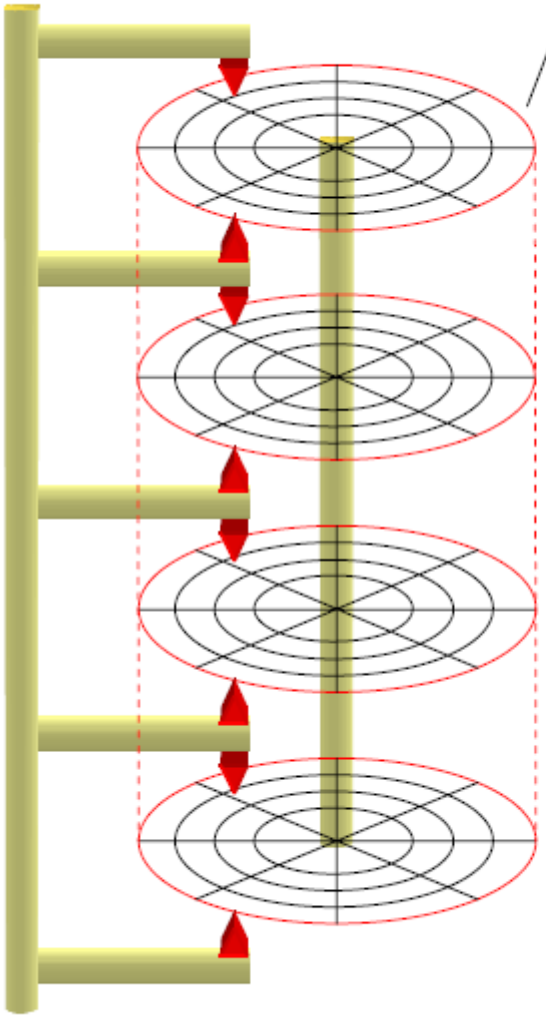
Setor

Cada trilha é divida em setores. Uma operação de leitura/gravação atua sobre um setor inteiro



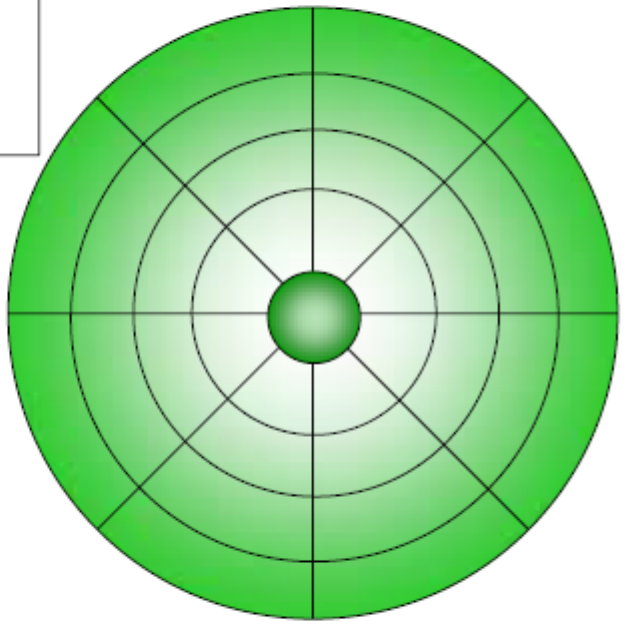
Setor

Terminologia sobre Discos

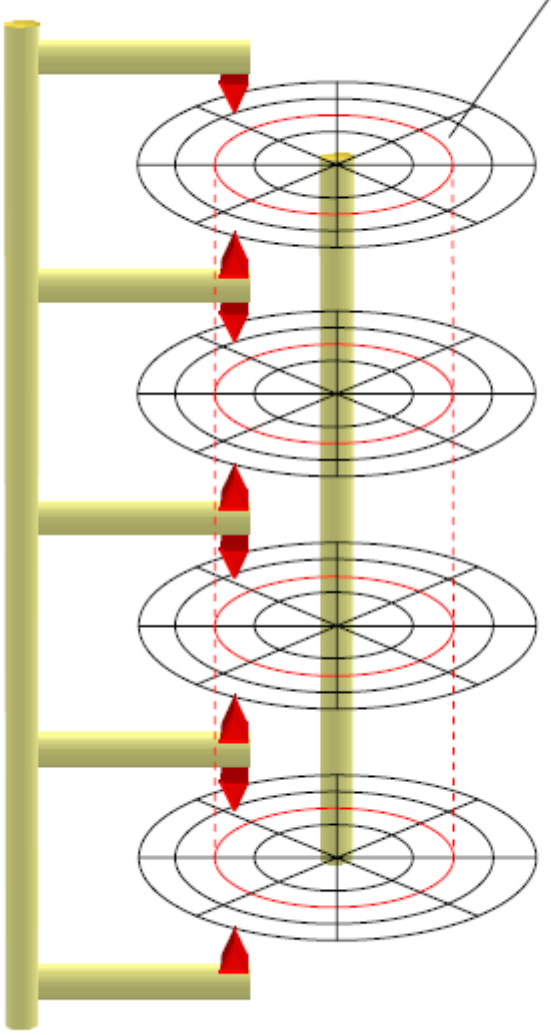


Cilindro

Todas as trilhas que ficam em uma mesma posição do braço de leitura/gravação

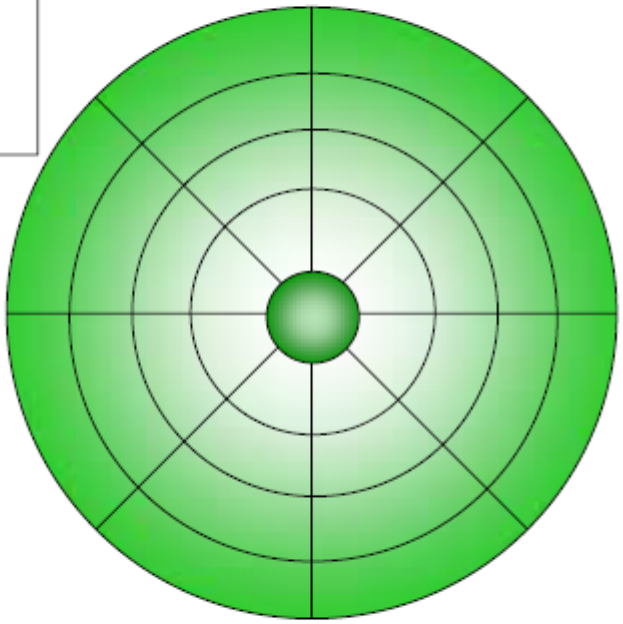


Terminologia sobre Discos

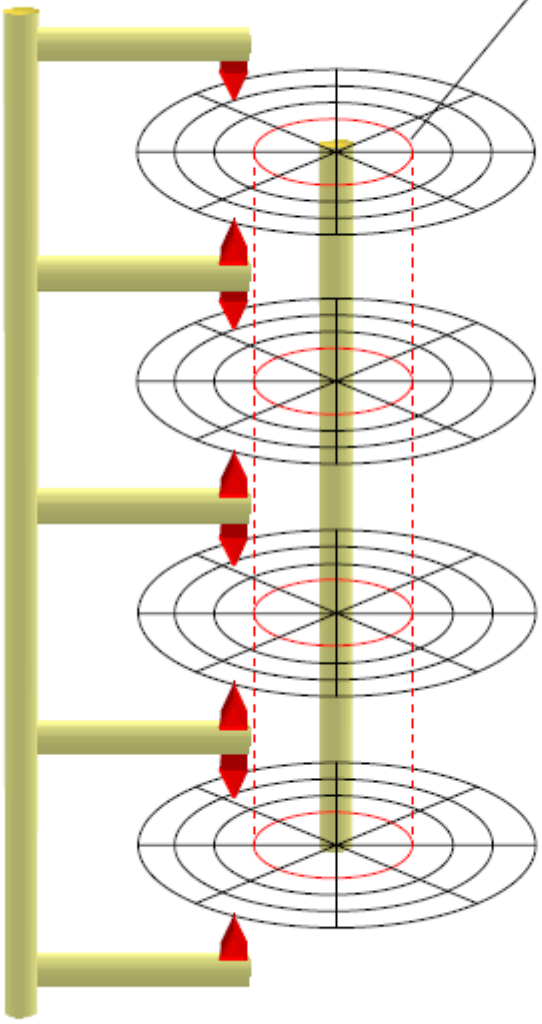


Cilindro

Todas as trilhas que ficam em uma mesma posição do braço de leitura/gravação

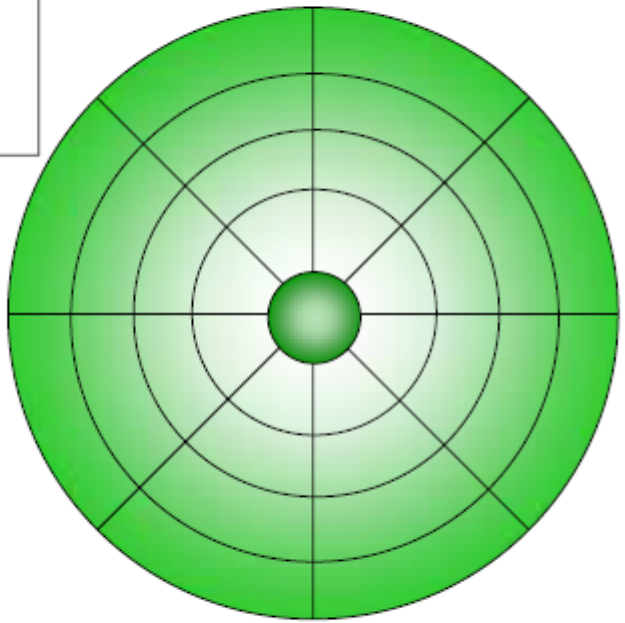


Terminologia sobre Discos



Cilindro

Todas as trilhas que ficam em uma mesma posição do braço de leitura/gravação



Terminologia sobre Discos

Um mecanismo impulsor de discos tem um eixo rotor onde são montados os discos e um conjunto de cabeçotes (ou cabeças) de leitura/gravação

Existe uma cabeça de leitura/gravação para cada superfície

Durante a leitura ou gravação, os cabeçotes estão estacionários sobre os discos em posição onde a leitura ou gravação deve ser efetuada, enquanto os discos giram em alta velocidade

Assim, o dispositivo vai ler ou gravar em círculos concêntricos de cada superfície

Terminologia sobre Discos

- A área que pode ser lida ou gravada por uma única cabeça estacionada, é chamada de uma *trilha*
- As trilhas são então círculos concêntricos e cada vez que o disco completa uma rotação, uma trilha completa passa em frente do cabeçote da leitura/gravação
- A coleção de trilhas de todas as superfícies que se encontram simultaneamente sob os cabeçotes de leitura/gravação é chamada de *cilindro*
- As trilhas são divididas em setores
- Um *setor* é o menor segmento endereçável de uma trilha
- As informações são gravadas por blocos, ao longo das trilhas de uma superfície

Terminologia sobre Discos

- Para usar um disco deve-se especificar o número de trilha, ou cilindro, o número do setor no qual começa o bloco, como também a superfície
- Posiciona-se primeiro o conjunto de cabeçotes para o cilindro correto
- Antes de começar a leitura/gravação, espera-se a chegada do setor indicado abaixo do cabeçote de leitura/gravação
- Ao término disso, pode ser feita a transmissão de dados

Terminologia sobre Discos

- Assim, existem três fatores que contribuem para o tempo de entrada/saída no caso de discos:
- (i) **Tempo de posicionamento** (*seek time*): tempo para posicionar os cabeçotes de leitura/ gravação para o cilindro (ou trilha) correto; depende da quantidade de cilindros (trilhas) que os cabeçotes precisam se deslocar (movimento mecânico do braço)
- (ii) **Tempo de latência** (*latency time*): tempo decorrido até o que setor correto da trilha chegue abaixo do cabeçote de leitura/gravação (espera rotacional)
- (iii) **Tempo de transmissão***:* tempo para transmitir o bloco de dados de/para o disco

Terminologia sobre Discos

- Os tempos máximos de posicionamento em um disco são da ordem de 10 ms
- A rotação típica para discos é 7200 rpm, portanto, o tempo máximo de latência é no máximo 8.3 ms (tempo de uma rotação completa)
- As velocidades de transmissão ficam entre 10^5 bytes/s e 10^6 bytes/s
- A quantidade de bytes que podem ser gravados numa unidade de discos depende de números de superfícies e trilhas por superfície
- Esse valor varia de entre 10^9 bytes para discos pequenos até 10^{12} bytes para grandes unidades de discos

Tempo de Latência

- Tempo de latência
  - Pior caso, deve-se aguardar uma rotação completa = 60000/RPM ms
  - Caso médio, deve-se aguardar meia rotação = 30000/RPM ms
- Estas fórmulas são derivadas diretamente por regra de três, assumindo A rpm (Rotações por Minuto)
  - A rpm → 1 min → 60 s → 60000 ms
  - 1 rpm → x ms
  - x = 60000/A ms

Tempo de Latência

Velocidade (rpm)	Pior Caso (ms)	Caso Médio (ms)
5400	11.1	5.6
7200	8.3	4.2
10000	6.0	3.0
12000	5.0	2.5
15000	4.0	2.0

Tempo de Transferência

- Não é possível calcular o tempo de transmissão a partir dos outros fatores do disco
- Entretanto, uma maneira de aproximar esse valor consiste em saber quantos bytes passam sob as cabeças de leitura/gravação em um segundo, ou seja, a taxa de transferência de dados
- Isso depende da densidade dos dados (distância entre os bits em cada setor) como também da velocidade do disco
- A densidade pode ser calculada se o número de setores por trilha for conhecido (SPT), uma vez que há 512 bytes de dados em um setor
- A aproximação consiste em (em Megabits/s):
  - Taxa de Transferência de Dados = (RPM/ 60 * SPT * 512 * 8) / 1000000 Mbits/s
  - Para obter o resultado em megabytes/s basta dividir por 8

Exemplo

- Por exemplo, um disco 7200 rpm
  - Tempo de posicionamento = 10 ms
  - Tempo de latência = 8.3 ms
- Se o disco possui 407 setores por trilha, então sua taxa de transferência é de 200 Mbits/s = 25 Mbytes/s = 25000 bytes/ms
- Assim, o tempo de leitura de um único setor é de 512/25000 = 0.02048 ms
- Para um arquivo com 100 registros, cada registro ocupando 300 bytes, alocados em setores distintos o tempo de leitura é
  - 100 * (10 + 8.3 + 0.02048) = 1832.048 ms ~ 2 s

Sistema de Arquivos

- O Sistema de Arquivos (*file system*) é um software responsável por organizar setores do disco em arquivos e diretórios
- O Sistema de Arquivos gerencia quais setores pertencem a um determinado arquivo bem como quais setores não estão sendo utilizados
- Assim, o Sistema de Arquivos permite trabalhar com arquivos de forma lógica, sem se preocupar com os detalhes do armazenamento físico

Sistema de Arquivos

- O Sistema de Arquivos (*file system*) é um software responsável por organizar setores do disco em arquivos e diretórios
- O Sistema de Arquivos gerencia quais setores pertencem a um determinado arquivo bem como quais setores não estão sendo utilizados
- Assim, o Sistema de Arquivos permite trabalhar com arquivos de forma lógica, sem se preocupar com os detalhes do armazenamento físico

Arquivos

- Em nosso contexto, um arquivo é definido como um conjunto de **registros**
- Os registros são formados por unidades de informação denominadas **campos**
- Em geral, há um campo especial denominado **chave**, que identifica univocamente cada registro
- Cada arquivo possui associado um **ponteiro de arquivo** que indica a posição (registro) do arquivo onde será efetuada a próxima leitura/gravação
- Os registros são numerados seqüencialmente a partir da unidade (1, 2, 3, ...)
  - Portanto, um arquivo vazio possui zero registros

Arquivos

- Por exemplo, os registros de um arquivo de funcionários poderiam incluir os seguintes campos:
  - Número de matrícula (chave)
  - Nome
  - Cargo
  - Grau de escolaridade
  - Sexo (M,F)
  - Local
  - Estado civil (S, C)
  - Salário

Arquivos

- No Exemplo
  - Cada linha é um registro (5 registros)
  - Cada coluna é um campo (8 campos)
  - O ponteiro de arquivo (►) encontra-se no 3o. registro

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	800	Austin	programador	2	F	Campinas	S	10.000
2	510	William	analista	3	M	Campinhas	C	15.000
► 3	950	Melissa	analista	3	F	Vinhedo	S	12.000
4	750	Hawkins	programador	2	M	Campinas	S	12.000
5	620	Newton	programador	2	M	Vinhedo	C	9.000

Especificação

- Operações elementares em arquivos
  - abrir (*open*)
  - fechar (*close*)
  - ler (*read, input*)
  - escrever (*write, output*)
  - testar pelo final-de-arquivo (eof = *end-of-file*)
  - posicionar (*seek*) o arquivo em um determinado registro
  - encontrar a posição atual do ponteiro de arquivo (registro atual ou corrente)

Criação

File::File(string filename, Mode modo)

- *pré-condição:* o arquivo não esteja aberto
- *pós-condição:* o ADT é associado ao arquivo em disco de nome *filename* e o arquivo é aberto no modo especificado
  - enum Mode {read, write, readwrite};
- Em geral, arquivos abertos nos modos **read** ou **readwrite** devem existir em disco; no modo **write** todo conteúdo do arquivo é descartado (se existir) e o arquivo é aberto completamente sem registros (vazio)

Destruição

File::~File();

- *pré-condição:* O arquivo tenha sido criado
- *pós-condição:* O arquivo é fechado, garantindo que toda informação seja efetivamente escrita em disco; após fechado, o arquivo correspondente do sistema operacional permanece residente em disco

Status

bool File::Eof();

- *pré-condição:* Arquivo tenha sido criado
- *pós-condição:* função retorna **true** se a próxima leitura/gravação será efetuada após o final do arquivo, ou seja, se o ponteiro de arquivo encontra-se no final do mesmo; **false** caso contrário

Operações Básicas

void File::Write(FileEntry x);

- *pré-condição:* Arquivo já tenha sido criado nos modos **write** ou **readwrite**
- *pós-condição:* O item **x** é armazenado na posição (registro) onde se encontra o ponteiro de arquivo, sobrescrevendo qualquer informação anterior; o ponteiro de arquivo avança para o próximo registro

Operações Básicas

void File::Write(FileEntry x);

- *pré-condição:* Arquivo já tenha sido criado nos modos **write** ou **readwrite**
- *pós-condição:* O item **x** é armazenado na posição (registro) onde se encontra o ponteiro de arquivo, s
  qualquer informação
  arquivo avança para

O tipo **FileEntry** depende da aplicação e pode variar desde um simples caracter ou número até uma **struct** ou **class** com muitos campos

Operações Básicas

void File::Read(FileEntry &x);

- *pré-condição:* Arquivo já tenha sido criado nos modos **read** ou **readwrite**
- *pós-condição:* O registro onde se encontra o ponteiro de arquivo é lido e copiado para a variável **x**; o ponteiro de arquivo avança para o próximo registro

Operações Básicas

void File::Seek(int n);

- *pré-condição:* Arquivo já tenha sido criado
- *pós-condição:* O ponteiro de arquivo é posicionado no registro de número **n**, sendo que qualquer operação de leitura/escrita subseqüente será efetuada no registro **n**

Outras Operações

int File::Size();

- *pré-condição:* Arquivo já tenha sido criado
- *pós-condição:* a função retorna o número de registros no arquivo

Outras Operações

int File::Pos();

- *pré-condição:* Arquivo já tenha sido criado
- *pós-condição:* a função retorna o registro atual (corrente) em que o arquivo se encontra

Pontos Importantes:

- Um arquivo pode aberto em três modos elementares:
  - Apenas para leitura (*read* ou *read only*)
  - Apenas para escrita (*write* ou *write only*)
  - Para leitura e escrita (*read-write*)
- O acesso aos registros em um arquivo pode ocorrer de forma
  - Seqüencial (um registro é lido/escrito após o anterior)
  - Aleatória (posiciona-se em um registro para posterior leitura/gravação) ou
  - uma combinação de ambas
- Como regra geral nas linguagens de programação, todo arquivo antes de ser utilizado deve ser **aberto**; ao término das operações com o arquivo ele deve ser **fechado**
- Em C++ o fechamento de arquivo é opcional, já que arquivos são objetos em C++ e seu *finalizador* se encarrega de fechar o arquivo quando o objeto deixa de existir; entretanto é uma boa prática de programação fechar um arquivo quando ele não é mais necessário para economizar recursos do sistema operacional

Organização de Arquivos

- O objetivo da organização de arquivos é proporcionar meios para a recuperação e atualização de registros
- A atualização de um registro pode incluir sua remoção, alteração de alguns dos seus campos ou a inserção de um registro completamente novo
- A recuperação de registros ocorre por meio de **consultas**

Tipos de Consultas

- Existem quatro tipos de consultas em arquivos:
  - Q1: Consulta simples: especifica-se o valor de um único campo
  - Q2: Consulta em intervalo: especifica-se um intervalo de valores para um único campo
  - Q3: Consulta funcional: especifica-se alguns valores determinados dentro do arquivo (exemplo: soma, média, desvio-padrão)
  - Q4: Consulta booleana: uma combinação booleana de Q1-Q3 utilizando os operadores lógicos **and**, **or** e **not**

Tipos de Consultas

- No exemplo, recupere os registros de todos os funcionários com:
  - Q1: Sexo = 'M'
  - Q2: Salário > 9000
  - Q3: Salário > média dos salários de todos os funcionários
  - Q4: (Sexo = 'F' **and** Cargo = 'Programador') **or** (Matricula > 700 **and** Sexo = 'M')

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	800	Austin	programador	2	F	Campinas	S	10.000
2	510	William	analista	3	M	Campinhas	C	15.000
3	950	Melissa	analista	3	F	Vinhedo	S	12.000
4	750	Hawkins	programador	2	M	Campinas	S	12.000
5	620	Newton	programador	2	M	Vinhedo	C	9.000

Tipos de Consultas

- No exemplo, recupere os registros de todos os funcionários com:
  - **Q1: Sexo = 'M'**
  - Q2: Salário > 9000
  - Q3: Salário > média dos salários de todos os funcionários
  - Q4: (Sexo = 'F' **and** Cargo = 'Programador') **or** (Matricula > 700 **and** Sexo = 'M')

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	800	Austin	programador	2	F	Campinas	S	10.000
2	**510**	**William**	**analista**	**3**	**M**	**Campinhas**	**C**	**15.000**
3	950	Melissa	analista	3	F	Vinhedo	S	12.000
4	**750**	**Hawkins**	**programador**	**2**	**M**	**Campinas**	**S**	**12.000**
5	**620**	**Newton**	**programador**	**2**	**M**	**Vinhedo**	**C**	**9.000**

Tipos de Consultas

- No exemplo, recupere os registros de todos os funcionários com:
  - Q1: Sexo = 'M'
  - **Q2: Salário > 9000**
  - Q3: Salário > média dos salários de todos os funcionários
  - Q4: (Sexo = 'F' **and** Cargo = 'Programador') **or** (Matricula > 700 **and** Sexo = 'M')

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	**800**	**Austin**	**programador**	**2**	**F**	**Campinas**	**S**	**10.000**
2	**510**	**William**	**analista**	**3**	**M**	**Campinhas**	**C**	**15.000**
3	**950**	**Melissa**	**analista**	**3**	**F**	**Vinhedo**	**S**	**12.000**
4	**750**	**Hawkins**	**programador**	**2**	**M**	**Campinas**	**S**	**12.000**
5	620	Newton	programador	2	M	Vinhedo	C	9.000

Tipos de Consultas

- No exemplo, recupere os registros de todos os funcionários com:
  - Q1: Sexo = 'M'
  - Q2: Salário > 9000
  - **Q3: Salário > média dos salários de todos os funcionários**
  - Q4: (Sexo = 'F' **and** Cargo = 'Programador') **or** (Matricula > 700 **and** Sexo = 'M')

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	800	Austin	programador	2	F	Campinas	S	10.000
2	**510**	**William**	**analista**	**3**	**M**	**Campinhas**	**C**	**15.000**
3	**950**	**Melissa**	**analista**	**3**	**F**	**Vinhedo**	**S**	**12.000**
4	**750**	**Hawkins**	**programador**	**2**	**M**	**Campinas**	**S**	**12.000**
5	620	Newton	programador	2	M	Vinhedo	C	9.000

Tipos de Consultas

- No exemplo, recupere os registros de todos os funcionários com:
  - Q1: Sexo = 'M'
  - Q2: Salário > 9000
  - Q3: Salário > média dos salários de todos os funcionários
  - **Q4: (Sexo = 'F' and Cargo = 'Programador') or (Matricula > 700 and Sexo = 'M')**

Registro	E#	Nome	Cargo	Esc.	Sexo	Local	E.C.	Salário
1	**800**	**Austin**	**programador**	**2**	**F**	**Campinas**	**S**	**10.000**
2	510	William	analista	3	M	Campinhas	C	15.000
3	950	Melissa	analista	3	F	Vinhedo	S	12.000
4	**750**	**Hawkins**	**programador**	**2**	**M**	**Campinas**	**S**	**12.000**
5	620	Newton	programador	2	M	Vinhedo	C	9.000