



Arquitetura e Organização de computadores

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves
danilo.alves@alu.ufc.br

MEMÓRIA PRIMÁRIA

- A memória é a parte do computador onde são armazenados programas e dados
 - Memória primária (ou memória principal) armazena programas de forma temporária, quando referidos pelo sistema operacional como **processos**



Bit

- A unidade básica de memória é o dígito binário, denominado bit (*binary digit*), que pode armazenar o **valor 0** ou o **valor 1**
- Memórias consistem em uma **quantidade de células** (ou locais), cada uma das quais podendo armazenar uma informação
 - A célula é a menor unidade endereçável



ENDEREÇOS DE MEMÓRIA

- Cada célula tem um número (endereço), pelo qual os programas podem se referir a ela
 - Se a memória possui n células, seus endereços serão $[0, n-1]$
 - Cada célula possui o mesmo número de bits (6, **8**, 10, 12, 16, ...)
 - Uma célula com k bits pode armazenar 2^k combinações diferentes de bits



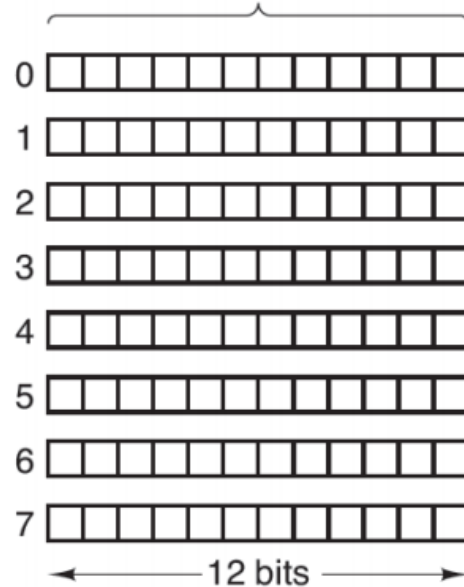
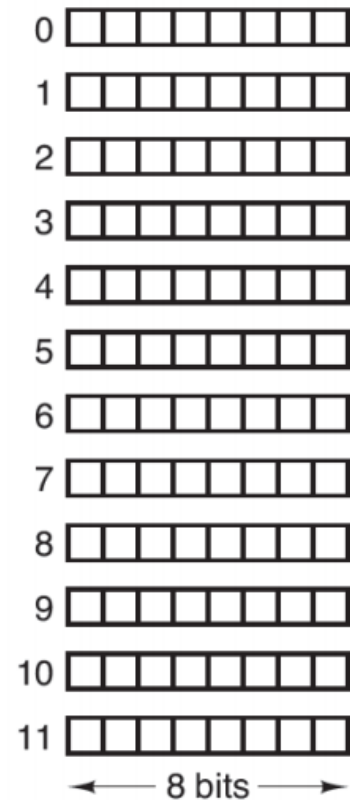
ENDEREÇOS DE MEMÓRIA



Endereço

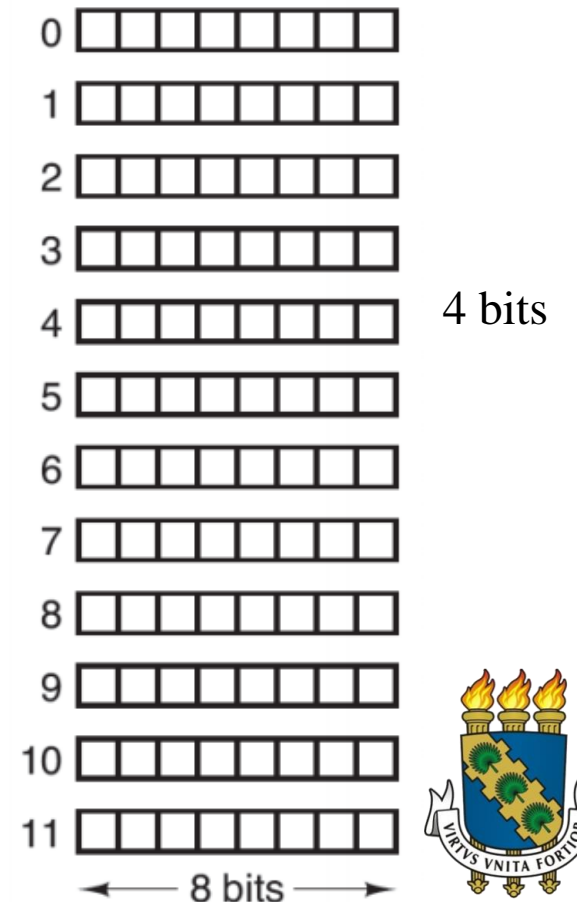
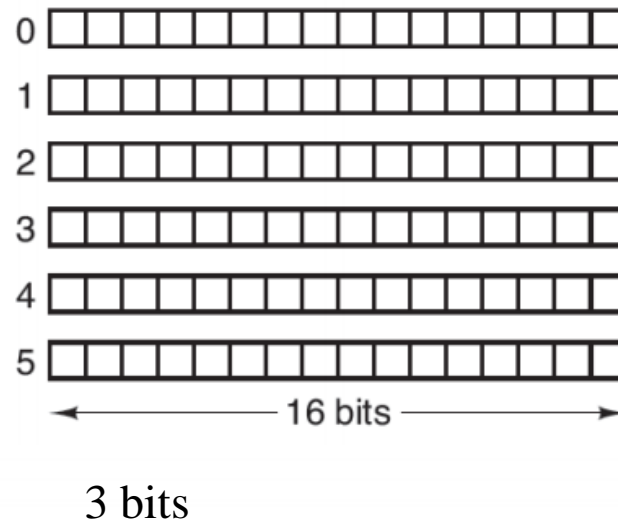
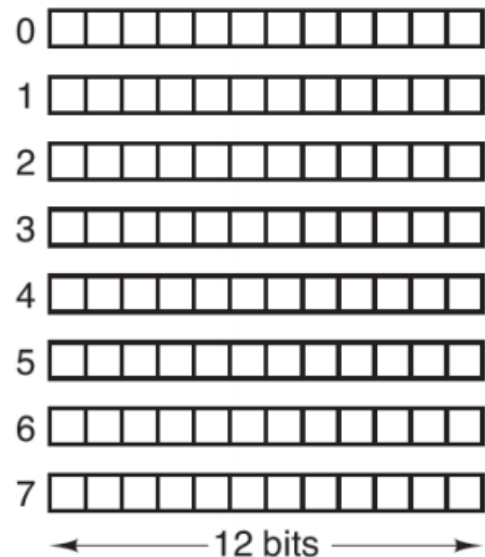
Endereço 1 célula

Endereço



ENDEREÇOS DE MEMÓRIA

- Computadores expressam endereços de memória como números binários.
- Se um endereço tiver m bits, o número máximo de células endereçáveis é 2^m
- Para a memória da figura precisa de no mínimo 3 e 4 bits



ENDEREÇOS DE MEMÓRIA

- O número de bits no endereço determina o número máximo de células diretamente endereçáveis na memória e é independente do número de bits por célula
 - Uma memória de 2^{12} células de 8 bits e uma memória com 2^{12} células de 64 bits cada precisam de endereços de 12 bits
- A célula é a menor unidade endereçável



ENDEREÇOS DE MEMÓRIA

- Bytes são agrupados em palavras (sequência de bits)
 - Um computador com uma palavra de 32 bits possui 4 bytes/palavra
 - Um computador com uma palavra de 64 bits possui 8 bytes/palavra
- A significância de uma palavra é que grande parte das instruções efetua operações com palavras inteiras
 - Máquina de 32 bits terá registradores de 32 bits e instruções para manipular palavras de 32 bits
 - Máquina de 64 bits terá registradores de 64 bits e instruções para manipular palavras de 64 bits
 - Movimentar, somar, subtrair, etc

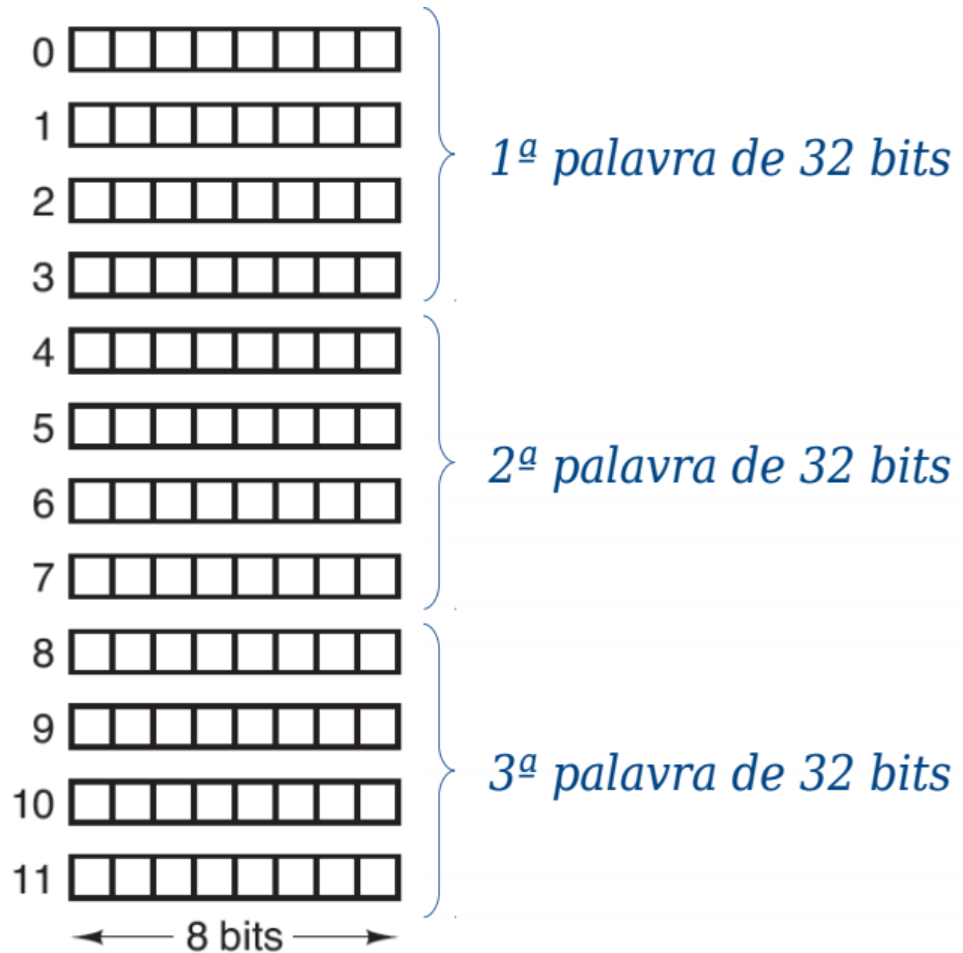


ENDEREÇOS DE MEMÓRIA

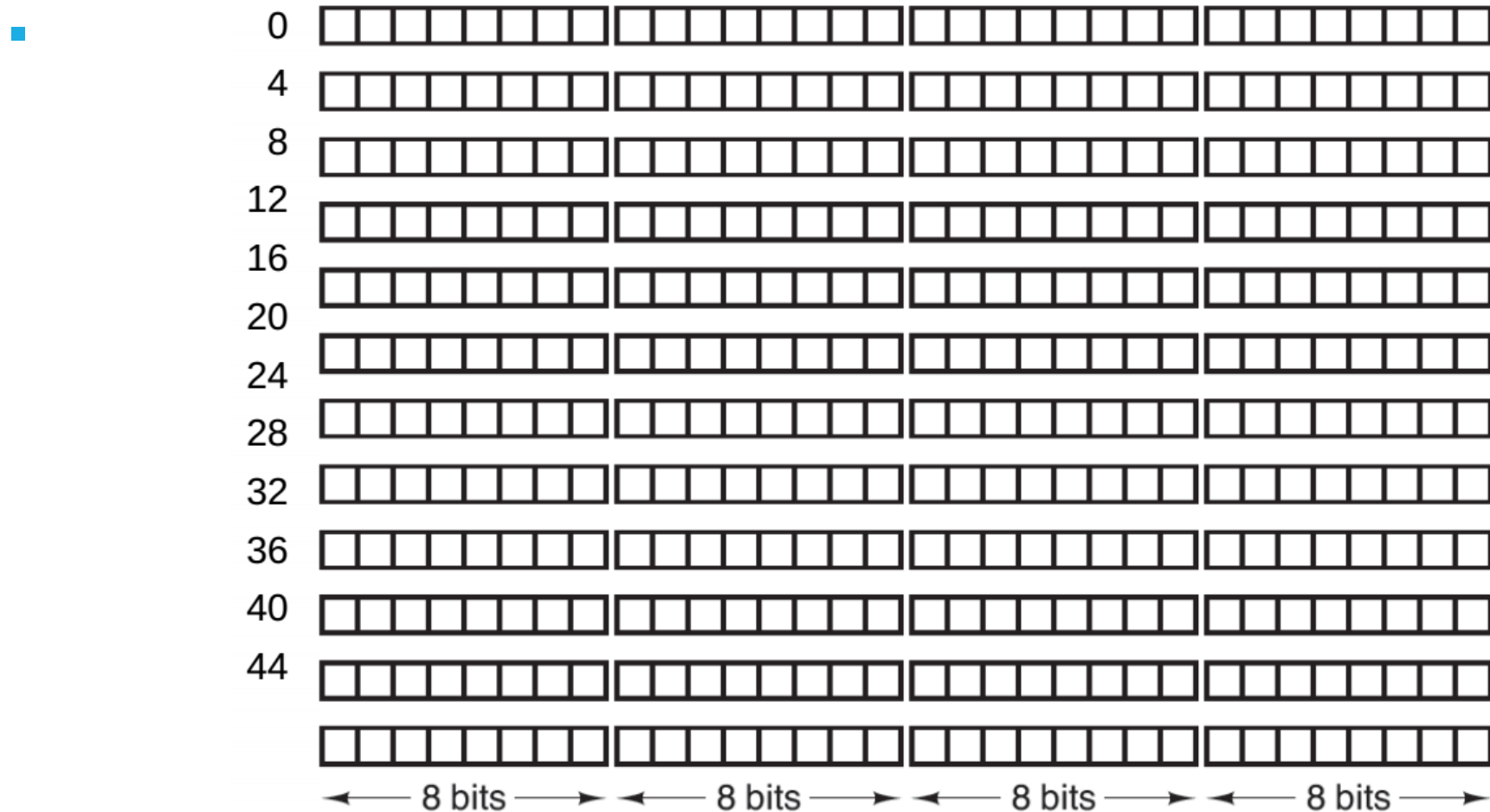
- Para simplificar, podemos fazer uma analogia: na arquitetura de 32 bits, enquanto um processador precisa realizar duas “viagens” (uma a cada ciclo de clock) para interpretar determinadas informações, na de 64 bits, ele realizaria apenas uma
- Dessa forma, a capacidade de um hardware do gênero poder trabalhar com uma quantidade maior de bits, não influenciará diretamente em sua velocidade de operação, mas em um melhor desempenho geral da plataforma
 - A velocidade eventualmente será menor, mas a quantidade de instruções por ciclo será maior



ORDENAÇÃO DE BYTES

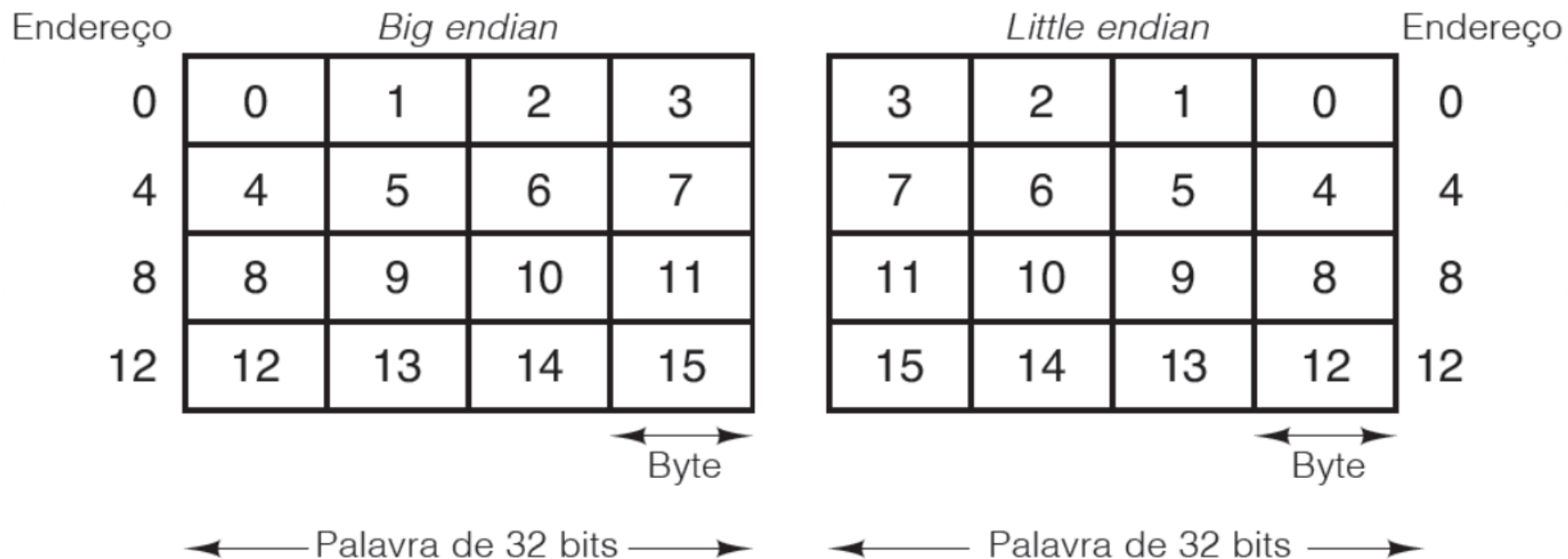


ORDENAÇÃO DE BYTES



ORDENAÇÃO DE BYTES

- Os bytes em uma palavra podem ser numerados da esquerda para a direita ou da direita para a esquerda
- Memória *big endian* e memória *little endian*



ORDENAÇÃO DE BYTES

- Suponha que tenhamos o valor hexadecimal de 32 bits 12345678 e que ele seja armazenado em uma palavra de 32 bits na memória endereçável por byte, no local de byte 184
- O valor consiste em 4 bytes, com o byte menos significativo contendo o valor 12345678 e o byte mais significativo contendo o valor 12345678

Endereço	Valor
184	12
185	34
186	56
187	78

Big endian

Endereço	Valor
184	78
185	56
186	34
187	12

Little endian



CÓDIGOS DE CORRECÇÃO DE ERRO

- Memórias de computador podem cometer erros de vez em quando devido a picos de tensão na linha eléctrica, por exemplo
- É preciso empregar métodos de detecção e até mesmo de correção de erros
 - Nesse caso, bits extras são adicionados às palavras de modo especial, para a validação
 - Ex.: A uma palavra de m bits de dados são adicionados r bits redundantes ($n = m + r$)



CÓDIGOS DE CORREÇÃO DE ERRO

- A partir de duas palavras, é possível determinar quantos bits são diferentes
 - Ex.: 10001001 e 10110001 → Há três bits diferentes
- O número de posições cujos bits são diferentes em duas palavras é a **distância de Hamming**
 - As propriedades de detecção de erro e correção de erro de um código dependem de sua distância de Hamming



CÓDIGOS DE CORREÇÃO DE ERRO

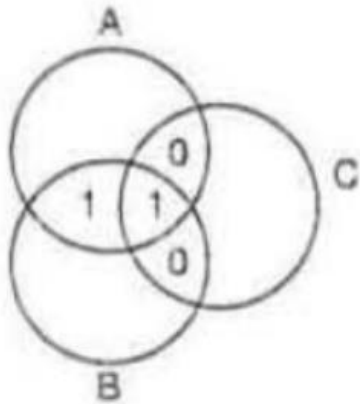
- Número de bits de verificação para um código que pode corrigir um erro único

Tamanho da palavra	Bits de verificação	Tamanho total	Acréscimo percentual
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2



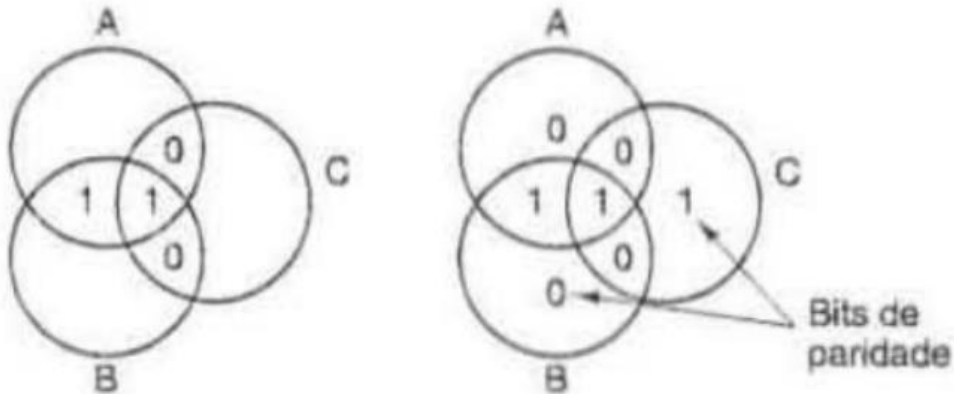
CÓDIGOS DE CORREÇÃO DE ERRO

- Codificar a palavra de 4 bits 1100 $\rightarrow AB = 1; ABC = 1; AC = 0; BD = 0$.



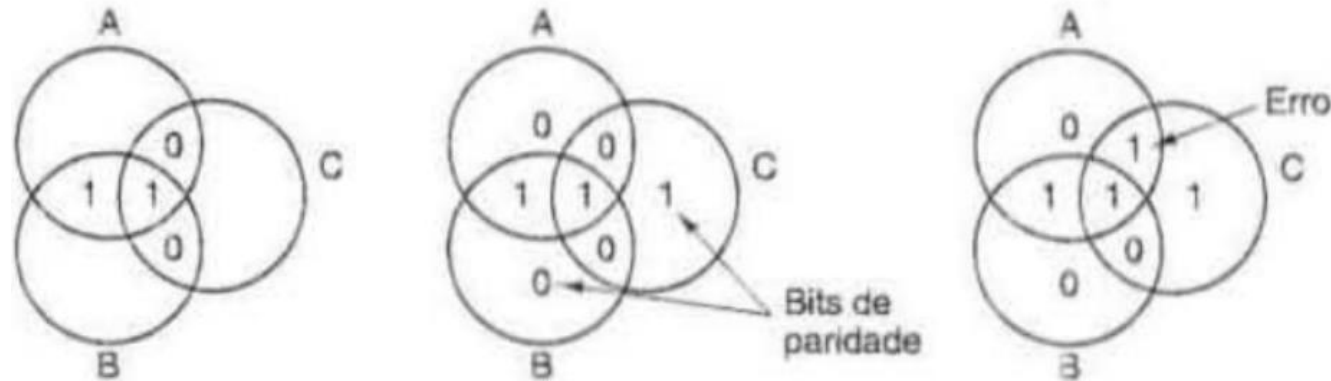
CÓDIGOS DE CORREÇÃO DE ERRO

- A, B e C armazenam bits de paridade par (ou ímpar)
 - A soma dos bits em cada círculo é par (ou ímpar)



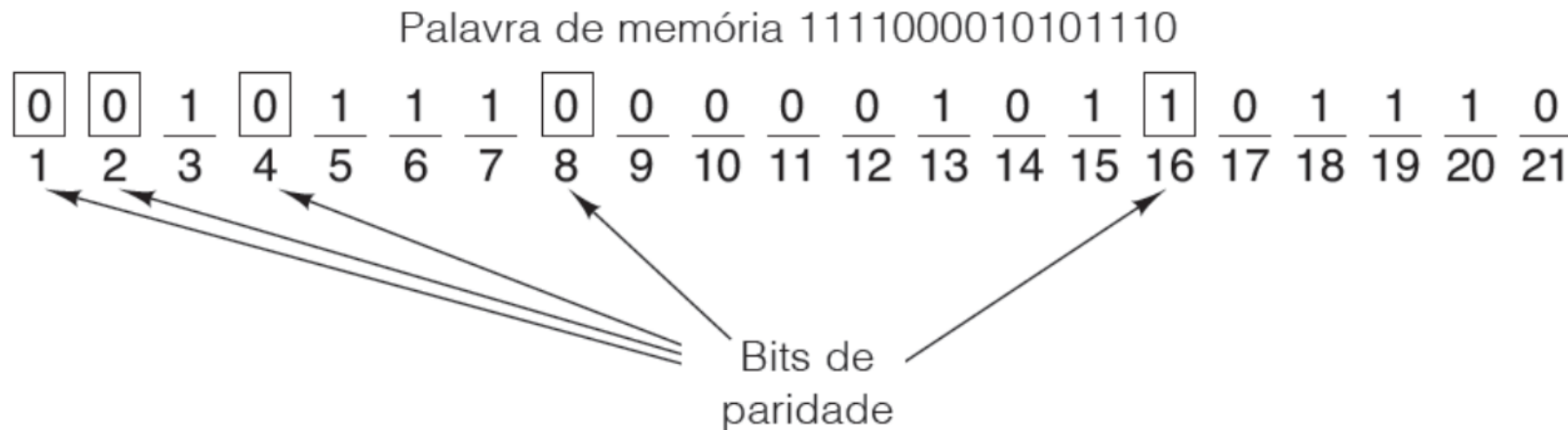
CÓDIGOS DE CORREÇÃO DE ERRO

- Se um bit da sequência 1100 for trocado para 1110, através dos bits de paridade é possível identificar que há um bit errado e qual o bit foi trocado, podendo ser feita sua correção



CÓDIGOS DE CORREÇÃO DE ERRO

- A figura a seguir mostra a construção de um código de Hamming para a palavra de memória de 16 bits 1111000010101110
- Todo bit potência de 2 é paridade
- A palavra de código de (16+5) 21 bits é 001011100000101101110
- Um bit b é verificado pelos bits de paridade ($b1, b2, b3, \dots$) de forma que $b1 + b2 + \dots = b$



CÓDIGOS DE CORREÇÃO DE ERRO

- A figura a seguir mostra a construção de um código de Hamming para a palavra de memória de 16 bits **1111000010101110**
- A palavra de código de $(16+5)$ 21 bits é **001011100000101101110**
- Para ver como funciona a correção de erros, considere o que aconteceria se o bit 5 fosse invertido por uma sobrecarga elétrica na linha de força
- A nova palavra de código seria **001001100000101101110**



CÓDIGOS DE CORREÇÃO DE ERRO

- Os bits são numerados começando de 1 (em vez de zero)

<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21



CÓDIGOS DE CORREÇÃO DE ERRO

- Bits cujas posições corresponderem à potência de 2 são de paridades

<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

- A palavra sem os códigos de paridade: 1 0 1 1 0 0 0 0 1 0 1 0 1 1 1 0

- Em seguida, fazemos as somas para cada bit de paridade



CÓDIGOS DE CORREÇÃO DE ERRO

- Os 5 bits de paridade serão verificados com os seguintes resultados:
 - Bit de paridade 1 incorreto (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contêm **cinco** 1s)
 - Bit de paridade 2 correto (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contêm seis 1s)
 - Bit de paridade 4 incorreto (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contêm **cinco** 1s)
 - Bit de paridade 8 correto (8, 9, 10, 11, 12, 13, 14, 15 contêm dois 1s)
 - Bit de paridade 16 correto (16, 17, 18, 19, 20, 21 contêm quatro 1s)



CÓDIGOS DE CORREÇÃO DE ERRO

- O bit incorreto deve ser um dos bits verificados pelo bit de paridade 1
 - 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 ou 21
- juntamente com um dos bits verificados pelo bit de paridade 4
 - 4, 5, 6, 7, 12, 13, 14, 15, 20 ou 21
- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos

1	3	5	7	9	11	13	15	17	19	21	
	4	5	6	7		12	13	14	15	20	21



CÓDIGOS DE CORREÇÃO DE ERRO

- Contudo, o bit 2 está correto

– 2, 3, 6, 7, 10, 11, 14, 15, 18, 19

- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos

1	3	5	7	9	11	13	15	17	19	21
	4	5	6		12	13	14		20	21



CÓDIGOS DE CORREÇÃO DE ERRO

- Da mesma forma, o bit 8 está correto

– 8, 9, 10, 11, 12, 13, 14, 15

- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos

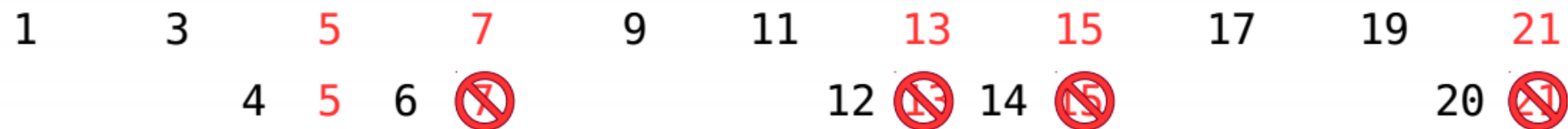
1	3	5	7	9	11	13	15	17	19	21
	4	5	6		12		14		20	21



CÓDIGOS DE CORREÇÃO DE ERRO

- Por fim, o bit 16 está correto
 - 16, 17, 18, 19, 20, 21

- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos











CÓDIGOS DE CORREÇÃO DE ERRO

- Restando apenas o bit 5. O que é necessário para corrigi-lo?

<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos

1	3	5		9	11			17	19			
	4	5				12		14			20	



CÓDIGOS DE CORREÇÃO DE ERRO

- Restando apenas o bit 5

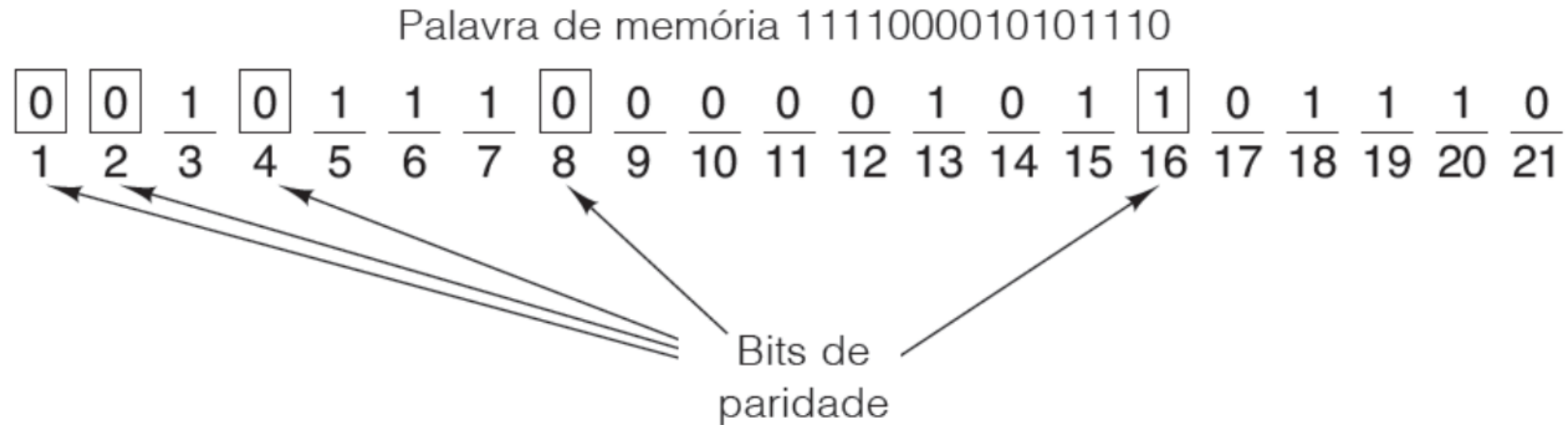
<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

- Como os bits de paridade inválidos são o 1 e o 4, o bit incorreto é verificado por ambos

1	3	5			9	11			17	19	
	4	5	6			12	14			20	



CÓDIGOS DE CORREÇÃO DE ERRO



CÓDIGOS DE CORREÇÃO DE ERRO

- Como são definidos os bits de paridade?

<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

- Bit de paridade 1 somam-se os bits 1 em (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21) $6\%2 = 0$
- Bit de paridade 2 somam-se os bits 1 em (2, 3, 6, 7, 10, 11, 14, 15, 18, 19) $6\%2 = 0$
- Bit de paridade 4 somam-se os bits 1 em (4, 5, 6, 7, 12, 13, 14, 15, 20, 21) $6\%2 = 0$
- Bit de paridade 8 somam-se os bits 1 em (8, 9, 10, 11, 12, 13, 14, 15) $2\%2 = 0$
- Bit de paridade 16 somam-se os bits 1 em (16, 17, 18, 19, 20, 21) $3\%2 = 1$



CÓDIGOS DE CORREÇÃO DE ERRO

- Determine os bits de paridade

—	—	<u>1</u>	—	<u>1</u>	<u>1</u>	<u>0</u>	—	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12

—	—	<u>1</u>	—	<u>0</u>	<u>0</u>	<u>1</u>	—	<u>0</u>	<u>1</u>	<u>1</u>	<u>0</u>
1	2	3	4	5	6	7	8	9	10	11	12

- Verificar se a palavra foi alterada

<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>1</u>
1	2	3	4	5	6	7	8	9	10	11	12

<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>0</u>	<u>0</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>
1	2	3	4	5	6	7	8	9	10	11	12



PROBLEMAS DE DESEMPENHO

- Historicamente, as CPUs sempre foram mais rápidas do que a memória
 - A medida que a memória melhora, a CPU também é aperfeiçoada, contribuindo para o desequilíbrio entre CPU e memória
 - Projetistas de memória costumam usar nova tecnologia para aumentar a capacidade
- Qual o significado desse desequilíbrio?
 - CPUs gastam muitos ciclos ociosas aguardando informações (instruções, dados) serem lidos/escritos da/na memória – quanto mais lenta a memória, mais ciclos a CPU terá de aguardar



PROBLEMAS DE DESEMPENHO

- É possível criar memórias que trabalhem tão rápidas quanto a CPU, mas para executarem, precisam se localizar dentro do chip da CPU – o uso dos barramentos contribui para a lentidão do acesso à memória
- Por que isso não é feito?
- Por fatores econômicos
 - Adicionar uma memória na CPU deixaria esta muito maior e mais cara
- Ou se tem uma memória pequena e rápida ou uma memória grande e lenta
- O que na prática precisamos é de muita memória a um preço baixo



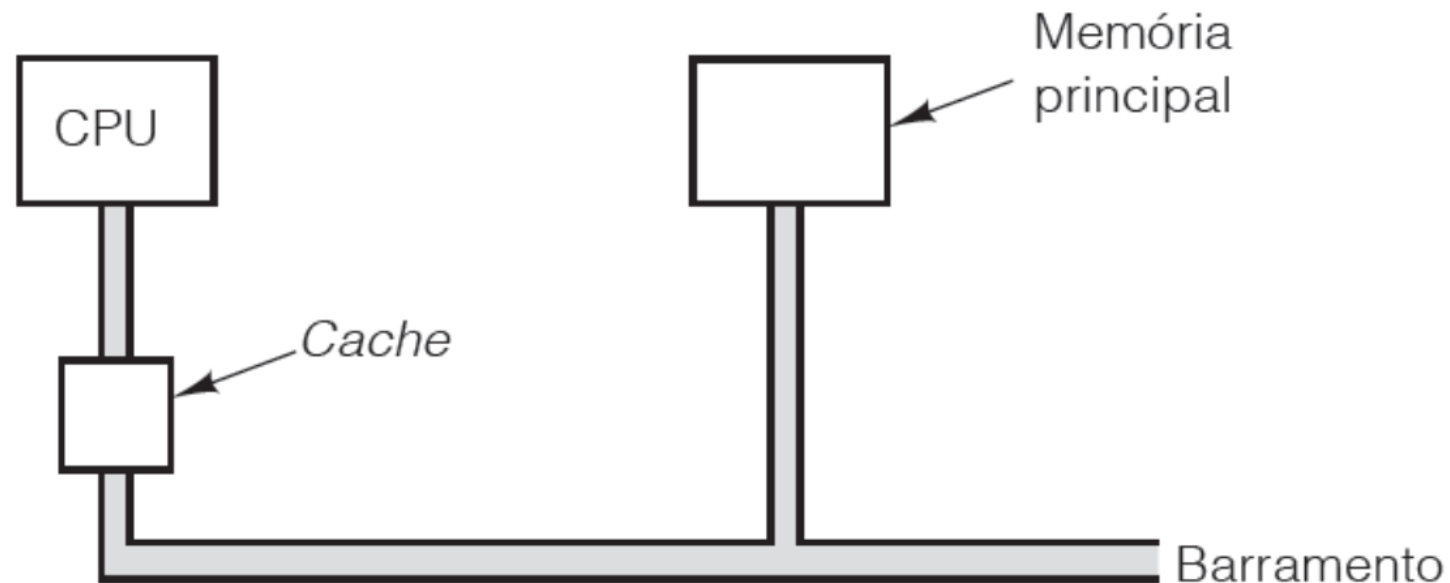
MEMÓRIA *CACHE*

- A ideia básica de uma cache é simples: as palavras de memória usadas com mais frequência são mantidas na cache
- Quando a CPU precisa de uma palavra, ela examina em primeiro lugar a cache.
- Somente se a palavra não estiver ali é que ela recorre à memória principal
- Se uma fração substancial das palavras estiver na cache, o tempo médio de acesso pode ser muito reduzido



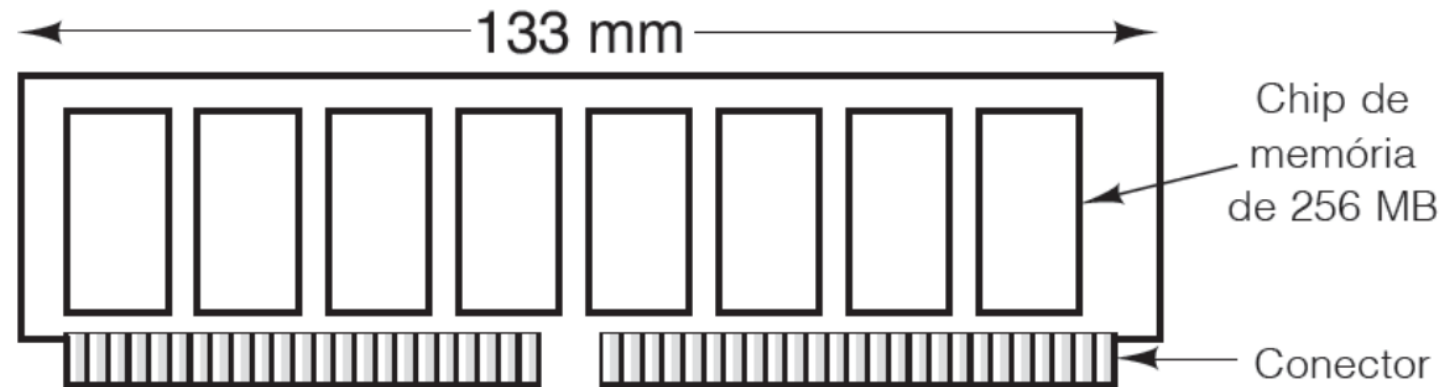
MEMÓRIA *CACHE*

- A localização lógica da cache é entre a CPU e a memória principal
- Em termos físicos, há diversos lugares em que ela poderia estar localizada



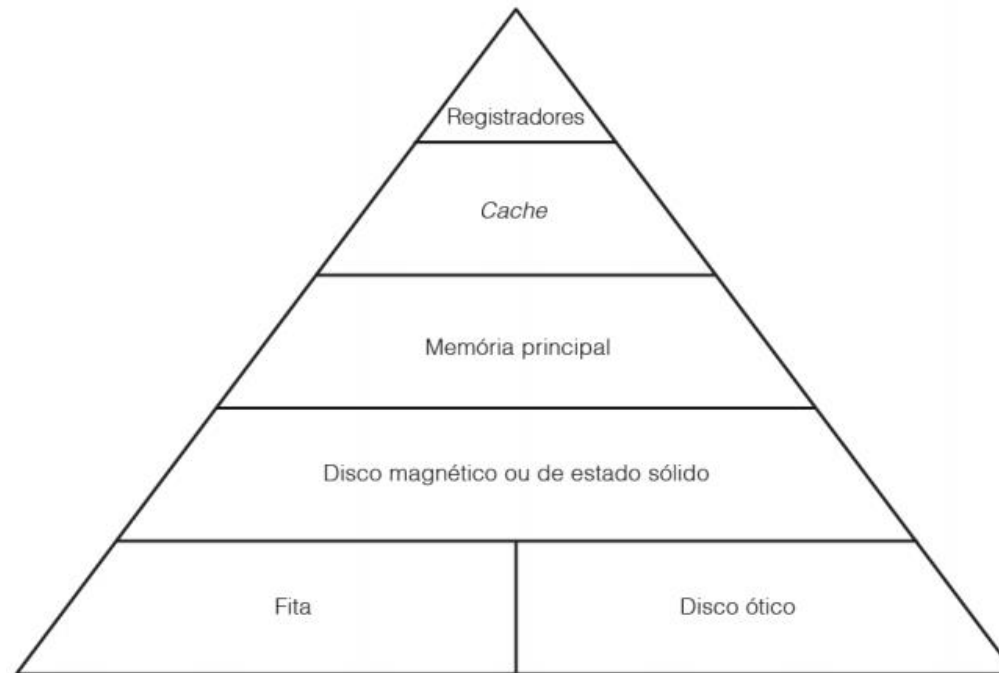
EMPACOTAMENTO E TIPOS DE MEMÓRIA

- Uma configuração típica de DIMM poderia ter oito chips de dados com 256 MB cada. Então, o módulo inteiro conteria 2 GB (ou 16 chips, totalizando 4GB)
- Muitos computadores têm espaço para quatro módulos, o que dá uma capacidade total de 8 GB se forem usados módulos de 2 GB, ou mais se forem usados módulos maiores



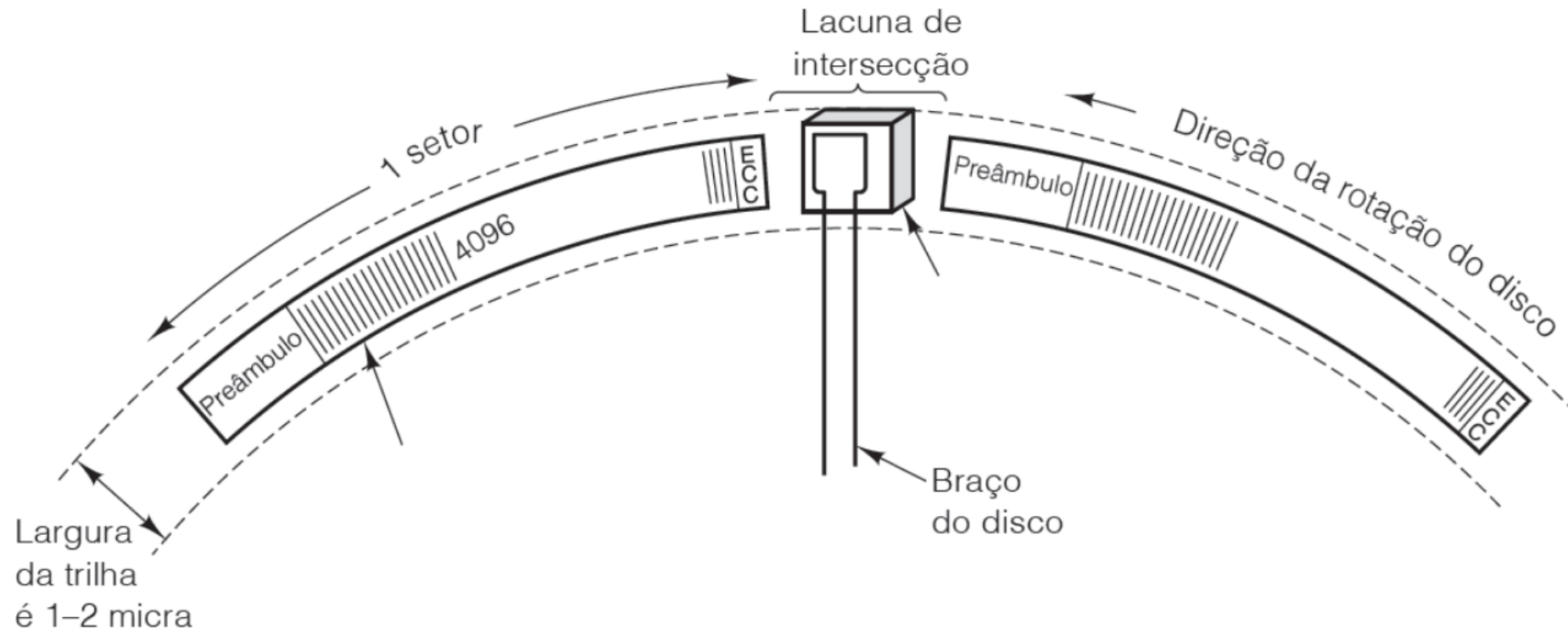
MEMÓRIA SECUNDÁRIA

- Seja qual for o tamanho da memória principal, ela sempre será muito pequena.
- A solução tradicional para armazenar grandes quantidades de dados é uma hierarquia de memória:



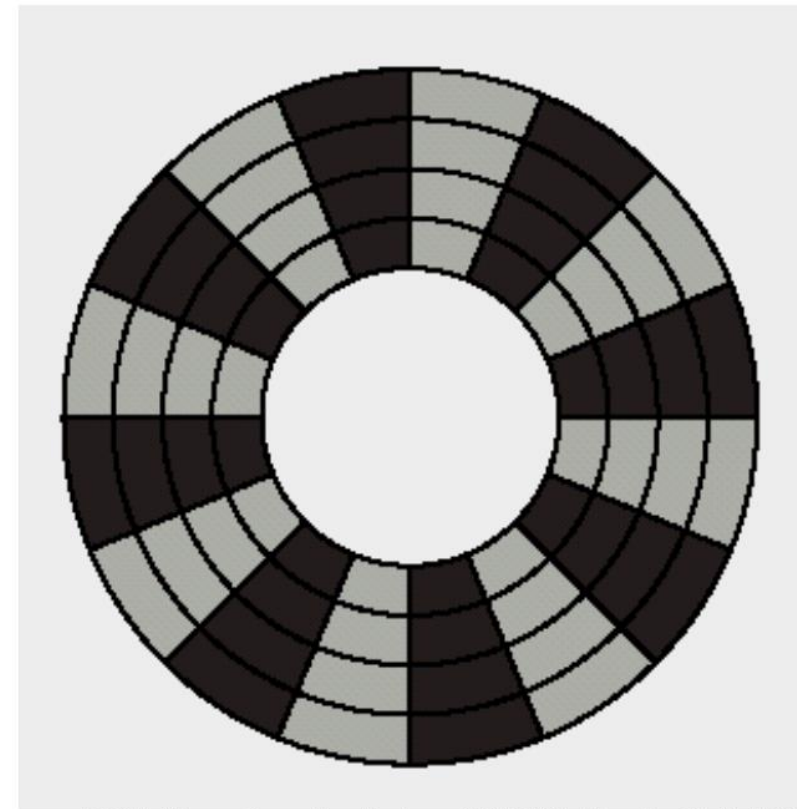
MEMÓRIA SECUNDÁRIA

- Um disco magnético é composto de um ou mais pratos de alumínio com um revestimento magnetizável



MEMÓRIA SECUNDÁRIA

- Duas tecnologias definem o número de bits por trilhas
 - O número de bits por trilha é constante
 - Trilhas mais internas possuem uma densidade maior bits/polegada
 - Discos com tecnologia CAV (*Constant Angular Velocity*)

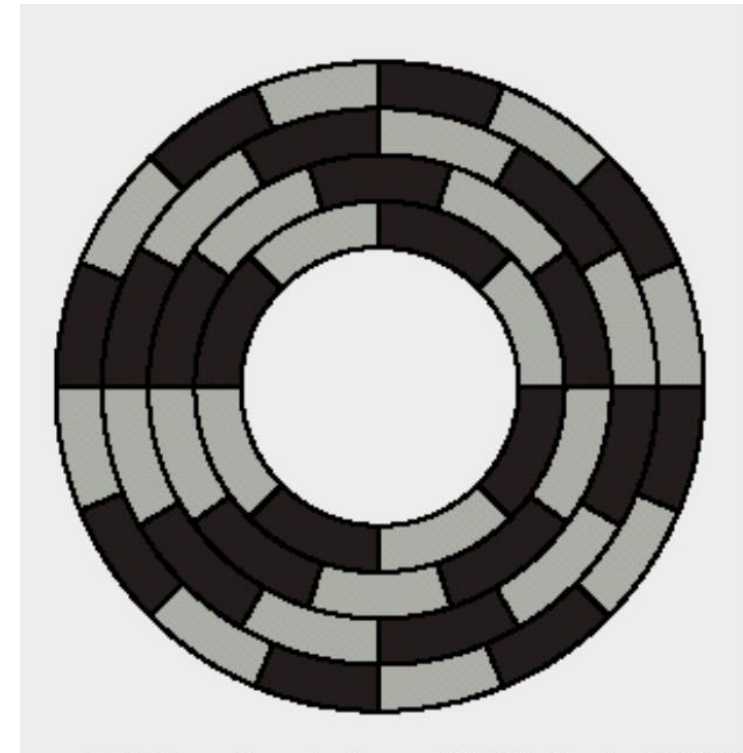


W. Stallings. *Operating Systems (4th Ed.)*, Prentice Hall, 2001.



MEMÓRIA SECUNDÁRIA

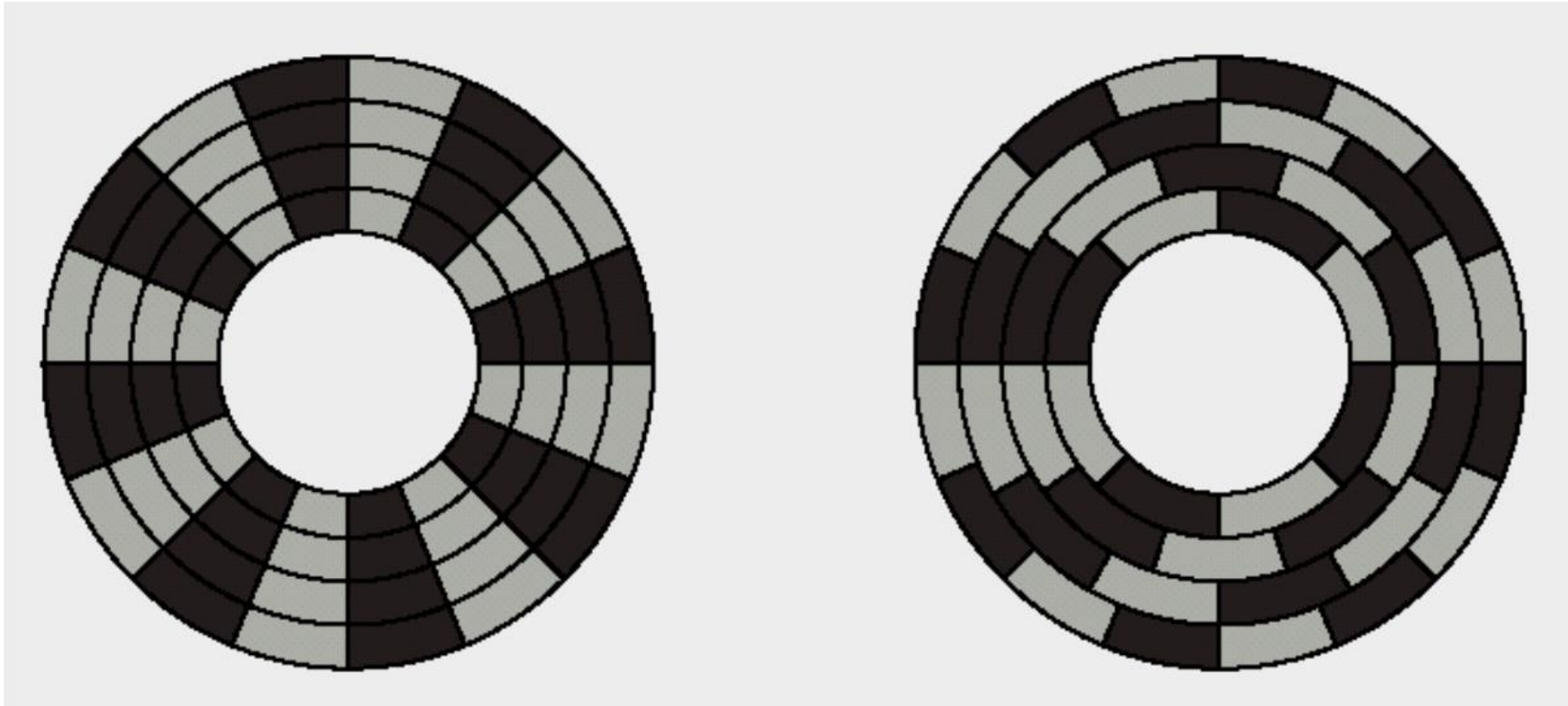
- Duas tecnologias definem o número de bits por trilhas
 - O número de bits por trilha depende se ela é mais interna ou mais externa
- Discos com tecnologia CLV (*Constant Linear Velocity*)
 - CDROM



W. Stallings. *Operating Systems (4th Ed.)*, Prentice Hall, 2001.



MEMÓRIA SECUNDÁRIA



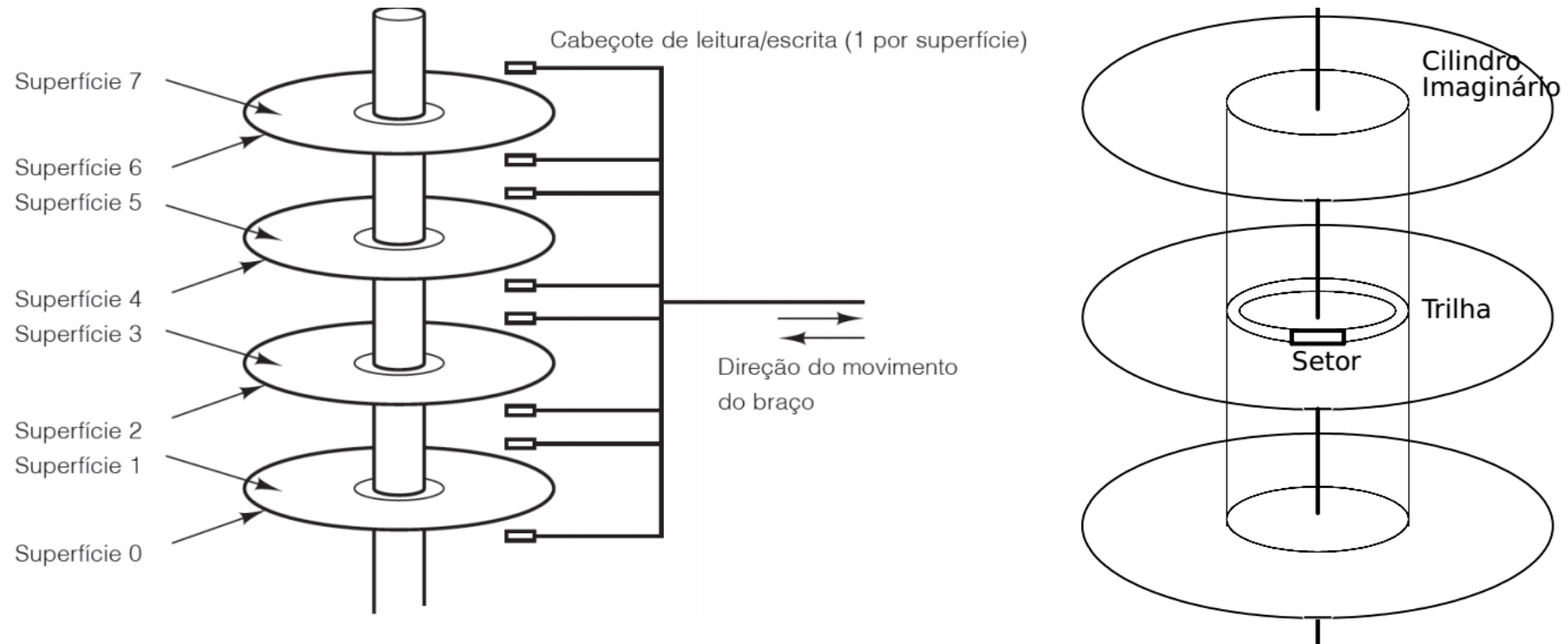
MEMÓRIA SECUNDÁRIA

- Múltiplos pratos (disk pack)
 - Vários pratos empilhados e centrados
 - Para cada prato, um cabeçote de leitura/escrita (braço móvel)
 - Cilindro: conjunto de trilhas de mesmo número em pratos diferentes



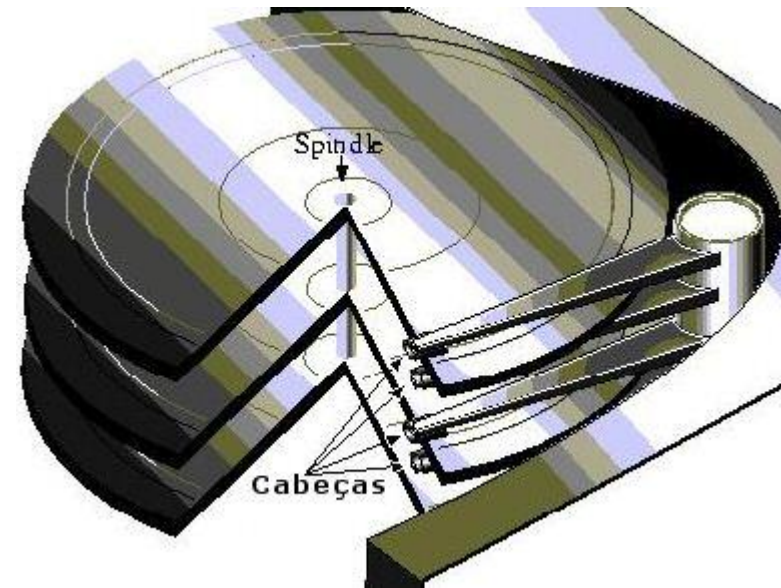
MEMÓRIA SECUNDÁRIA

- A maioria dos discos é composta de vários pratos empilhados na vertical:



MEMÓRIA SECUNDÁRIA

- A maioria dos discos é composta de vários pratos empilhados na vertical:

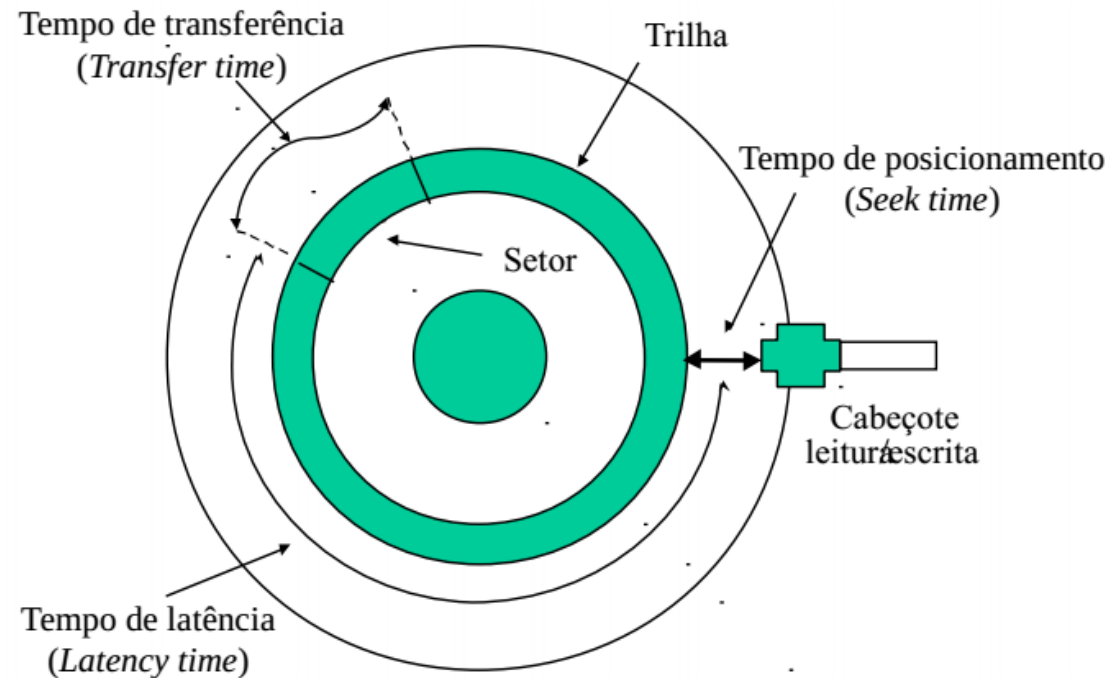


DESEMPENHO DO DISCO

- Para ler/escrever dados é necessário que o cabeçote de leitura e escrita esteja posicionada na trilha e no início do setor desejados
- Três tempos envolvidos:
 - Tempo de posicionamento (*seek time*)
 - Tempo necessário para posicionar o cabeçote de leitura/escrita na trilha desejada
 - Tempo de latência rotacional
 - Tempo necessário para atingir o início do setor a ser lido/escrito
 - Tempo de transferência
 - Tempo para escrita/leitura efetiva dos dados



TEMPORIZAÇÃO DO ACESSO AO DISCO



$$t_{\text{acesso}} = t_{\text{seek}} + t_{\text{latência}} + t_{\text{trasnf}}$$



TEMPO DE POSICIONAMENTO (*SEEK*)

- Possui duas componentes:
 - Tempo de acionamento e aceleração do braço do cabeçote
 - Tempo de deslocamento até a trilha desejada
- Não é linear em função do número do trilhas
 - Tempo de identificação da trilha (confirmação)
- Tempo médio de *seek*
 - Dado fornecido pelo fabricante
 - 5 a 10 ms (tecnologia ano 2000)



TEMPO DE LATÊNCIA ROTACIONAL

- Definido pela velocidade de rotação do motor
 - (ano 2000):
 - discos rígidos (5400 rpm a 10000 rpm)
 - unidades de floppy (300 rpm a 600 rpm)
- Considera-se o tempo médio
 - Não se sabe a posição relativa do cabeçote com a do setor a ser lido
 - Metade do tempo de uma rotação
 - 3 ms para um disco de 10000 rpm (6 ms uma rotação)



TEMPO DE TRANSFERÊNCIA

- Tempo de transferência de dados de/para disco depende da velocidade de rotação

$$T = \frac{b}{rN}$$

- T = tempo de transferência
- b = número de bytes a serem transferidos
- N = número de bytes em uma trilha
- r = velocidade de rotação, número de rotações por segundo



TEMPO MÉDIO DE ACESSO

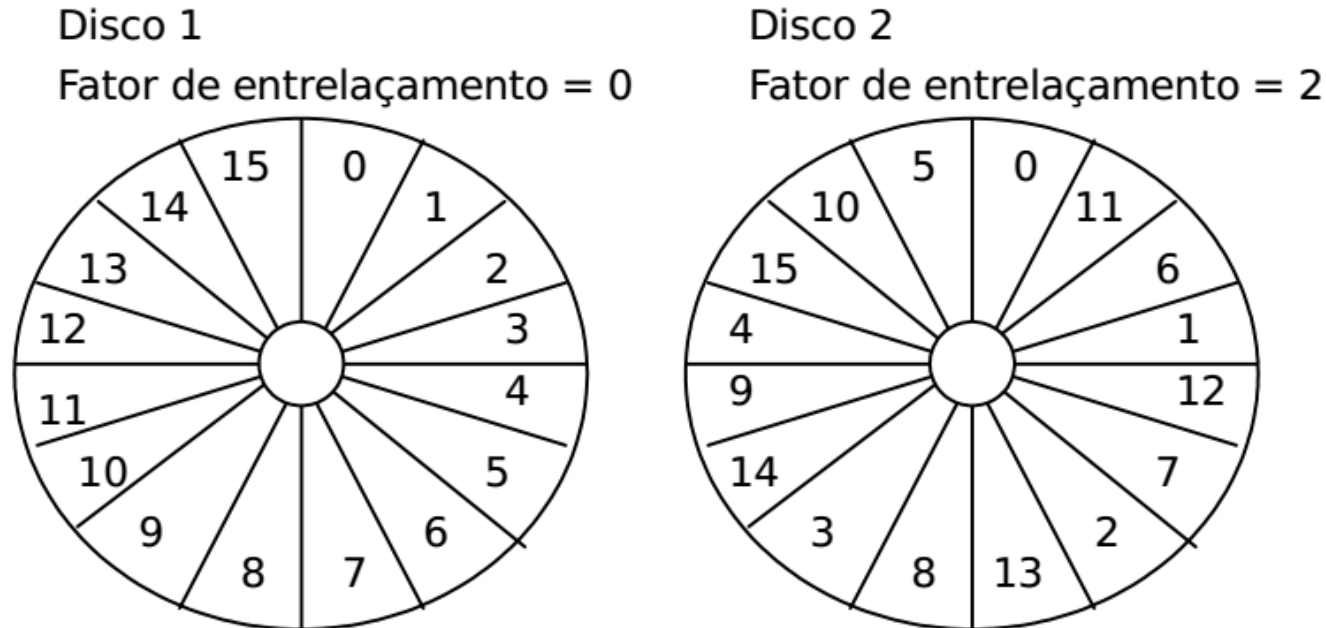
- Tempo médio de acesso é dado por:

$$T_{\text{acesso}} = t_{\text{seek_médio}} + \frac{1}{2r} + \frac{b}{rN}$$



ENTRELAÇAMENTO (INTERLEAVING)

- Forma de aumentar o desempenho no acesso ao disco
- Objetivo é evitar a latência rotacional em setores adjacentes
- Técnica consiste em numerar os setores não mais de forma contígua mas sim com um espaço entre eles



ESCALONAMENTO DE DISCO

- Tratar E/S em disco de forma eficiente se traduz em obter um tempo de acesso rápido e explorar ao máximo a largura de banda do disco
 - Traduz-se em minimizar o tempo de posicionamento (*seek*)
- Largura de banda do disco é definida como sendo o número total de bytes transferidos, divididos pelo tempo decorrido entre o primeiro acesso e a conclusão da transferência.



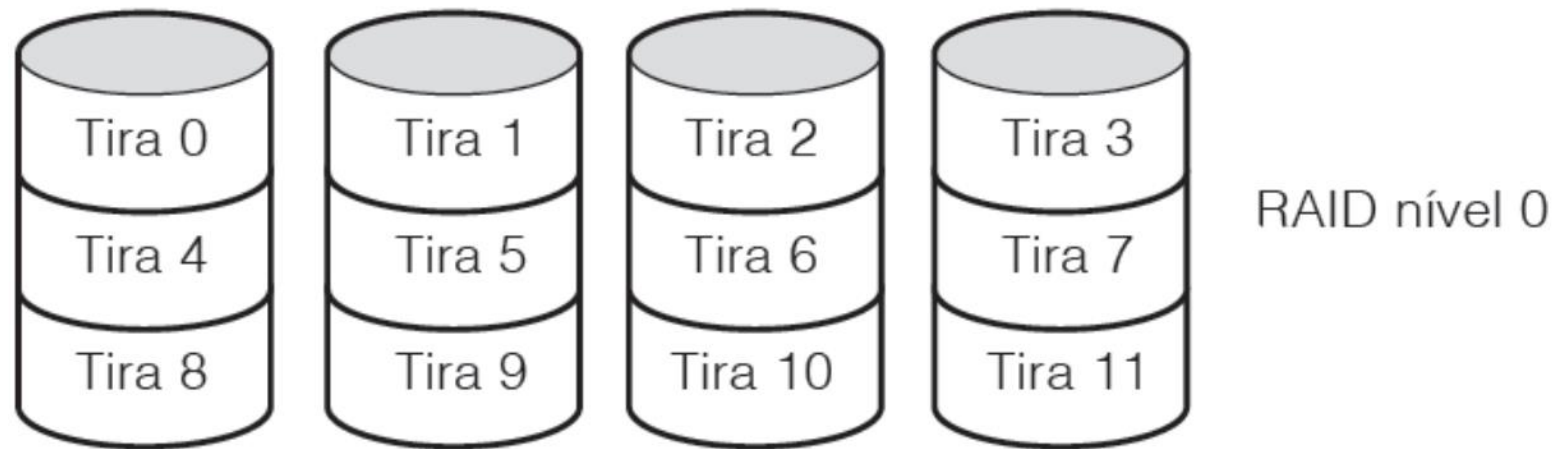
ESCALONAMENTO DE DISCO

- Algoritmos para reduzir o tempo de *seek*
 - Algoritmos de escalonamento do disco
 - Forma de organizar o atendimento a requisições
FCFS, SSTF, SCAN, C-SCAN, etc
- Exemplo para análise
 - Disco organizado em 200 trilhas (0-199)
 - Posição inicial do cabeçote: trilha 53
 - Atender a seguinte fila de requisições:
98, 183, 37, 122, 14, 124, 65, 67

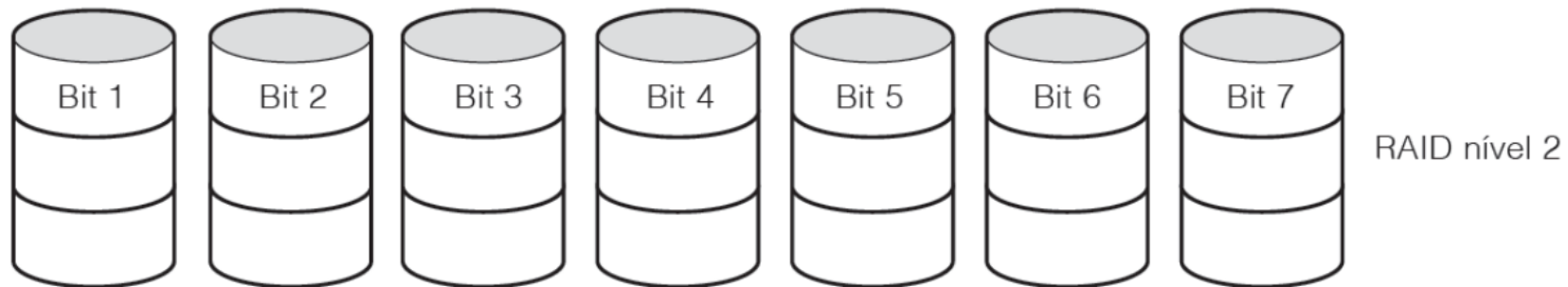
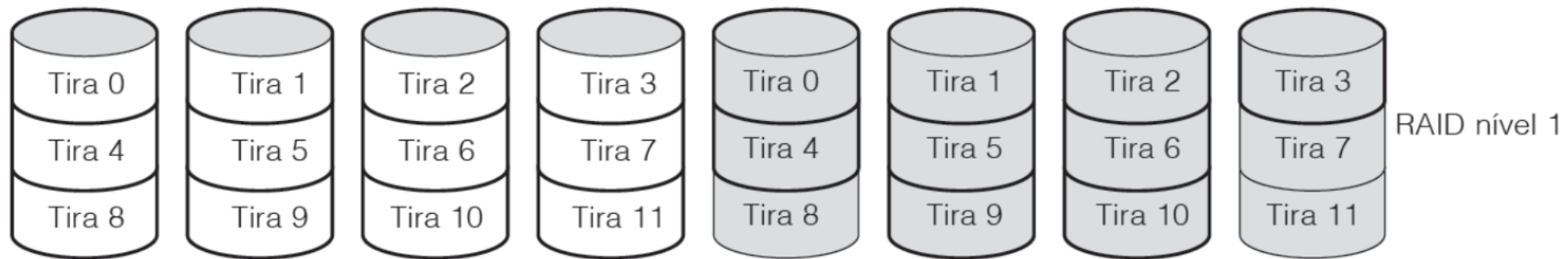


MEMÓRIA SECUNDÁRIA

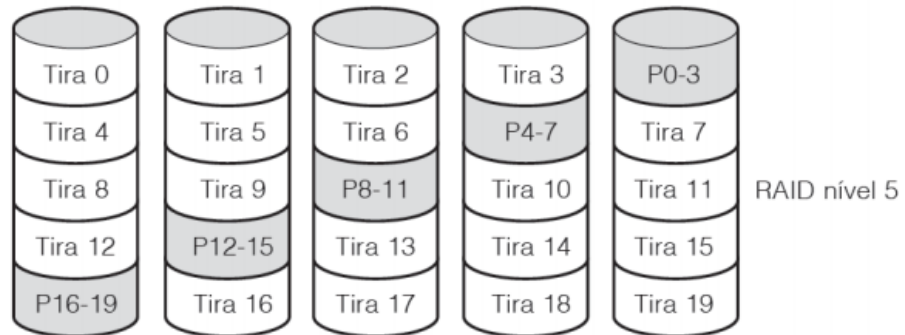
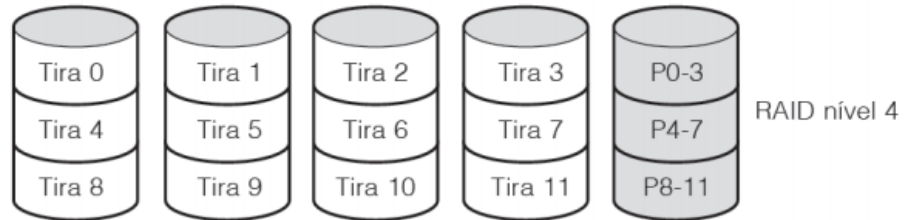
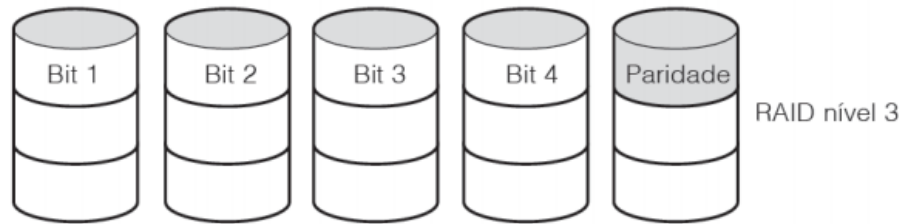
- Um RAID deveria parecer um SLED (*Single Large Expansive Disk*) para o sistema operacional, mas ter melhor desempenho e melhor confiabilidade
- RAIDs níveis 0 a 5. Os drives de backup e paridade estão sombreados



MEMÓRIA SECUNDÁRIA



MEMÓRIA SECUNDÁRIA

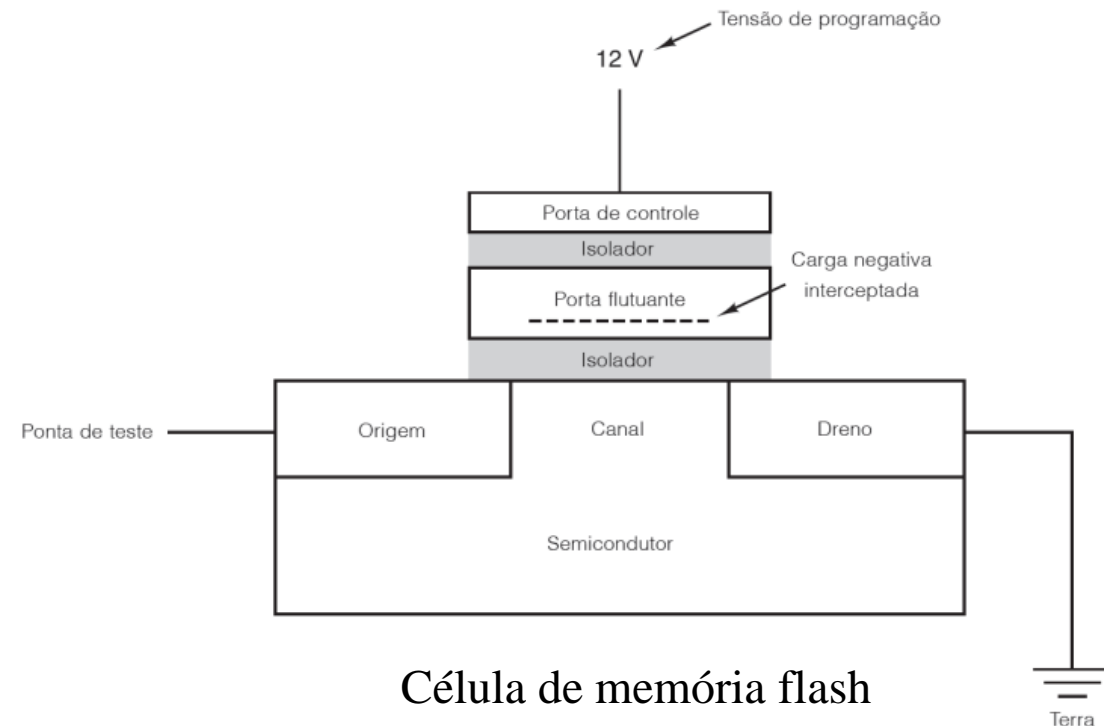


MEMÓRIA SECUNDÁRIA

- Discos feitos de memória flash não volátil, geralmente denominados discos em estado sólido
- Os discos flash são compostos de muitas células de memória flash em estado sólido



SSD



MEMÓRIA SECUNDÁRIA

- Vantagens do SSD em relação ao HD
 - O drive é 3 vezes menor, ou seja, ocupa pouco espaço
 - É totalmente silencioso e mais rápido (chega a ser 5 vezes mais veloz que um HDD)
 - É ideal para quem necessita de velocidade, pois carrega programas e arquivos rapidamente e melhora a performance do sistema
 - Não possui uma grande estrutura de discos mecânicos
 - É mais resistente em caso de queda, mas isso não quer dizer que seja indestrutível
 - Utiliza menor temperatura e menos consumo de eletricidade



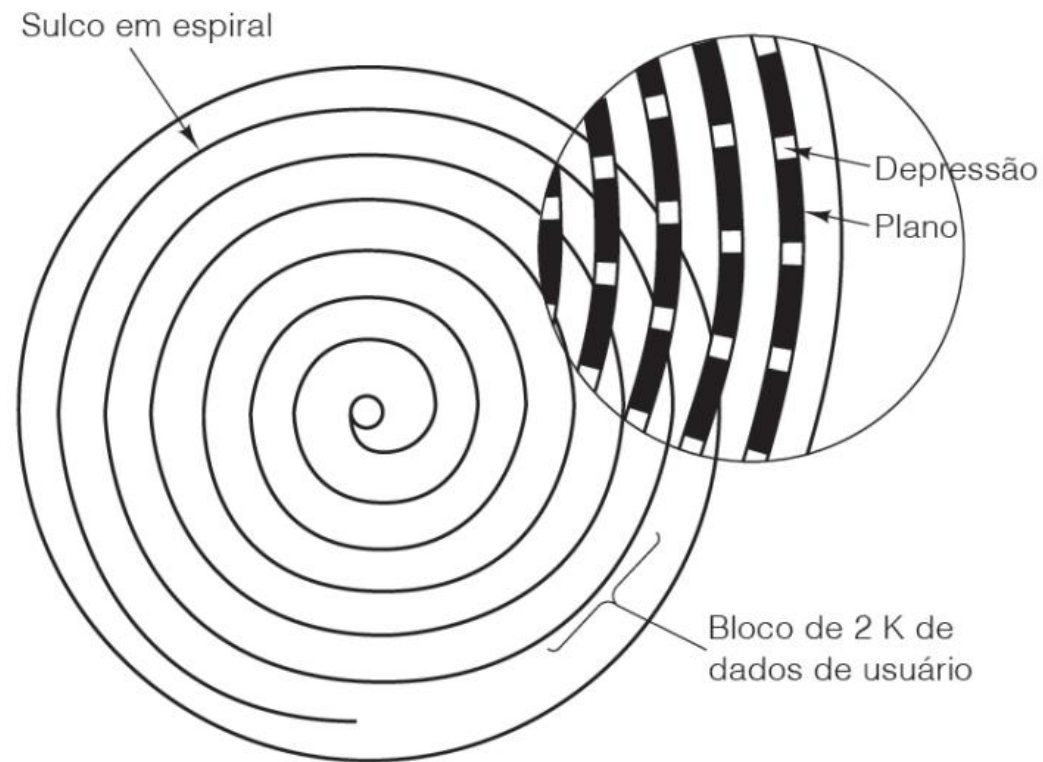
MEMÓRIA SECUNDÁRIA

- Desvantagens do SSD em relação ao HD
 - Preço alto, armazenamento baixo
 - Vida útil: toda vez que a memória recebe uma nova gravação (uma nova tensão elétrica), a célula vai perdendo um pouco a capacidade de segurar a carga elétrica. Depois de uma quantidade de vezes que isso ocorre, a memória “morre”.
 - Por ser uma tecnologia recente, é difícil de saber como será seu comportamento por um longo tempo



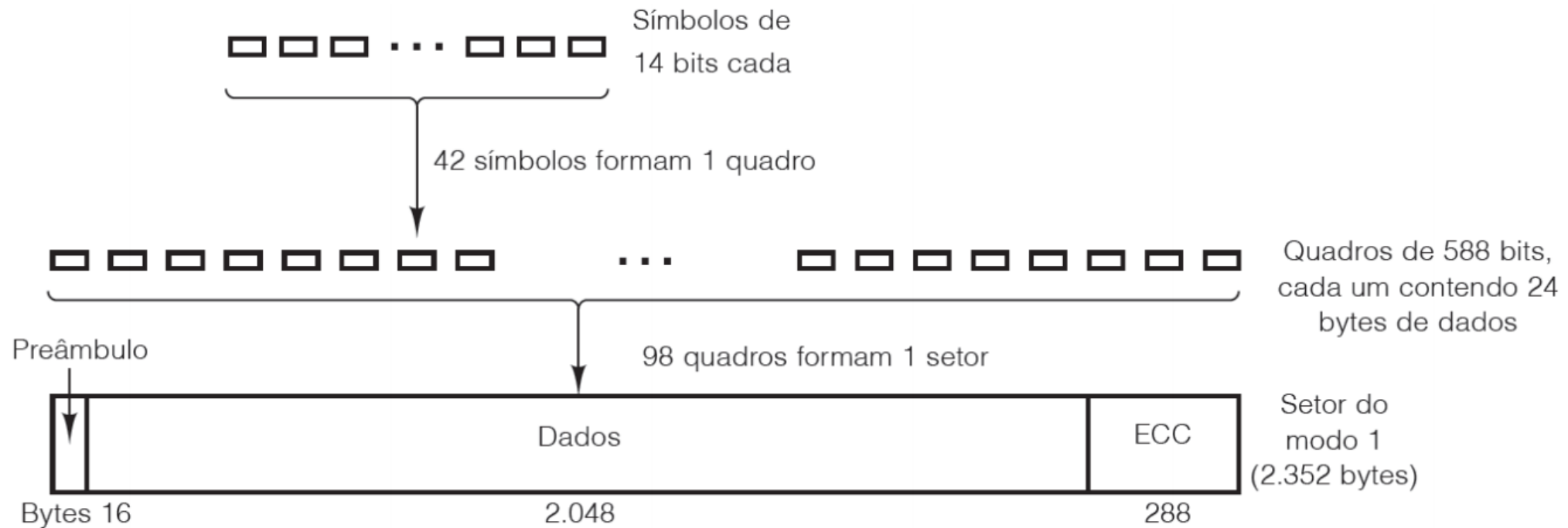
MEMÓRIA SECUNDÁRIA

- O formato básico de um CD-ROM consiste em codificar cada byte em um símbolo de 14 bits



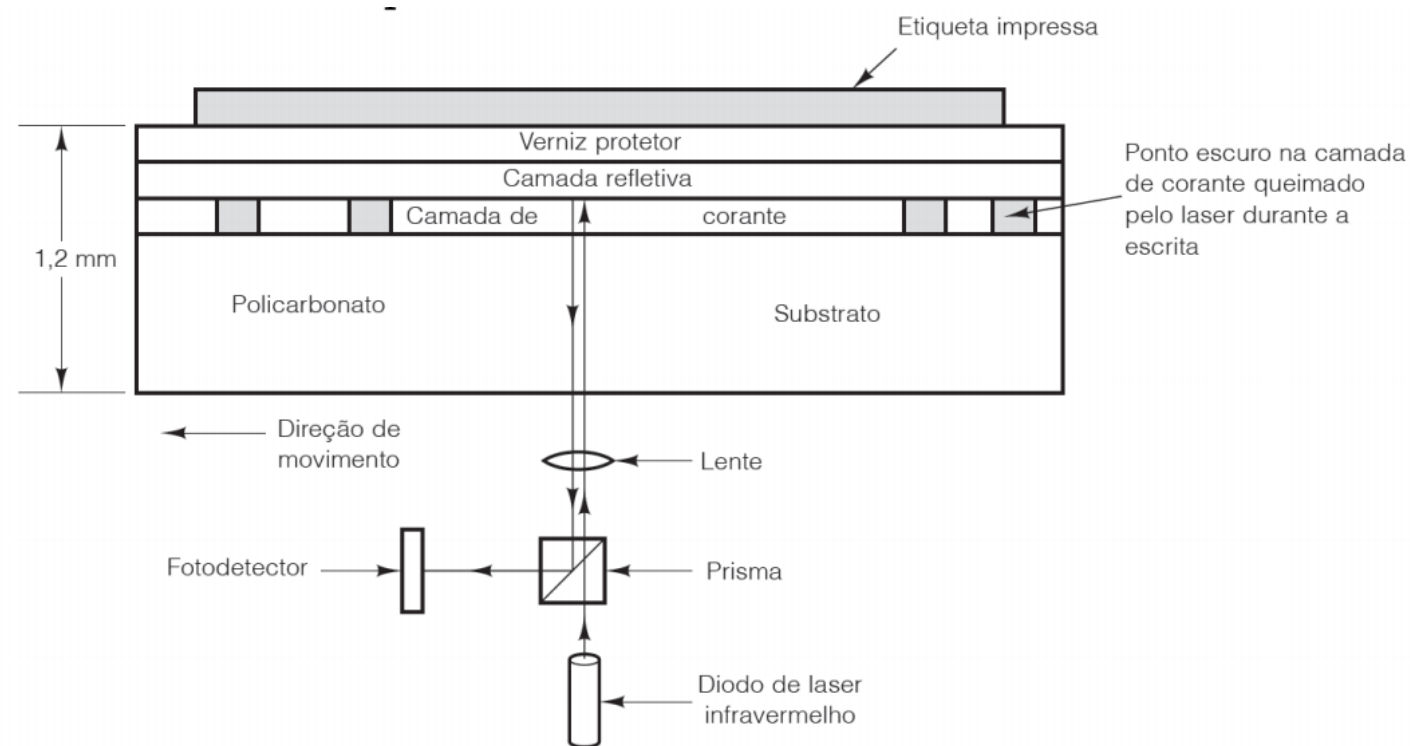
MEMÓRIA SECUNDÁRIA

- Leiaute lógico de dados em um CD-ROM



MEMÓRIA SECUNDÁRIA

- Nos CD-Rs as diferentes refletividades das depressões e dos planos têm de ser simuladas. Isso é feito com a adição de uma camada de corante entre o policarbonato e a superfície refletiva



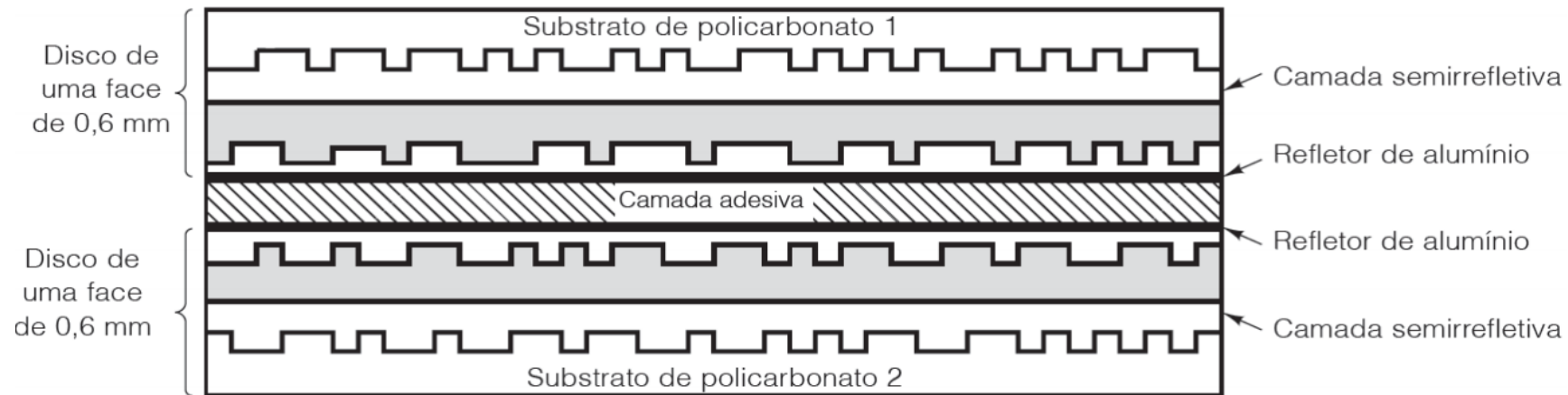
MEMÓRIA SECUNDÁRIA

- Uma tecnologia disponível agora é o **CD-RW** (CDs regraváveis), que usa um meio do mesmo tamanho do CD-R
- Contudo, o CD-RW usa uma liga de prata, índio, antimônio e telúrio para a camada de gravação
- Essa liga tem dois estados estáveis: cristalino e amorfo, com diferentes refletividades
- Uma combinação de tecnologia e demanda por três indústrias imensamente ricas e poderosas resultou no **DVD**



MEMÓRIA SECUNDÁRIA

- Disco de DVD de dupla face, dupla camada

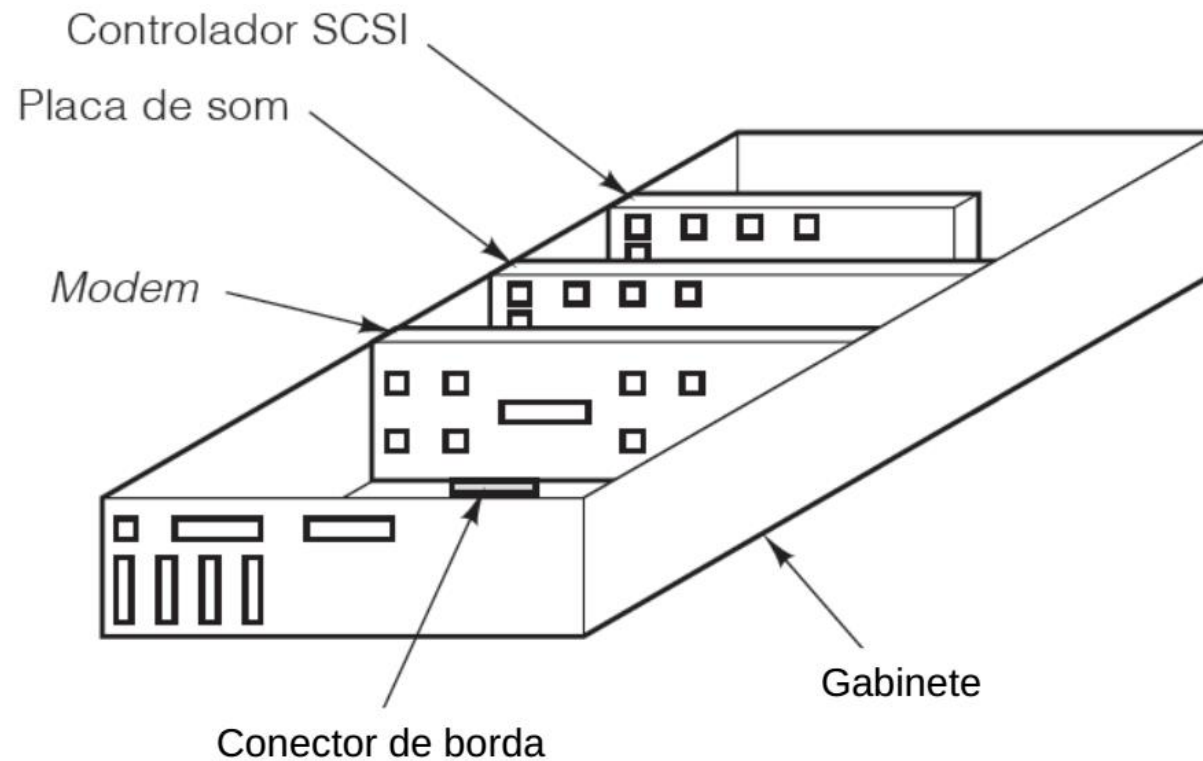


- O DVD mal acabara de ser lançado e seu sucessor já ameaçava torná-lo obsoleto
- O Blu-ray (raio azul), assim chamado porque usa um laser azul, em vez do vermelho usado por DVDs



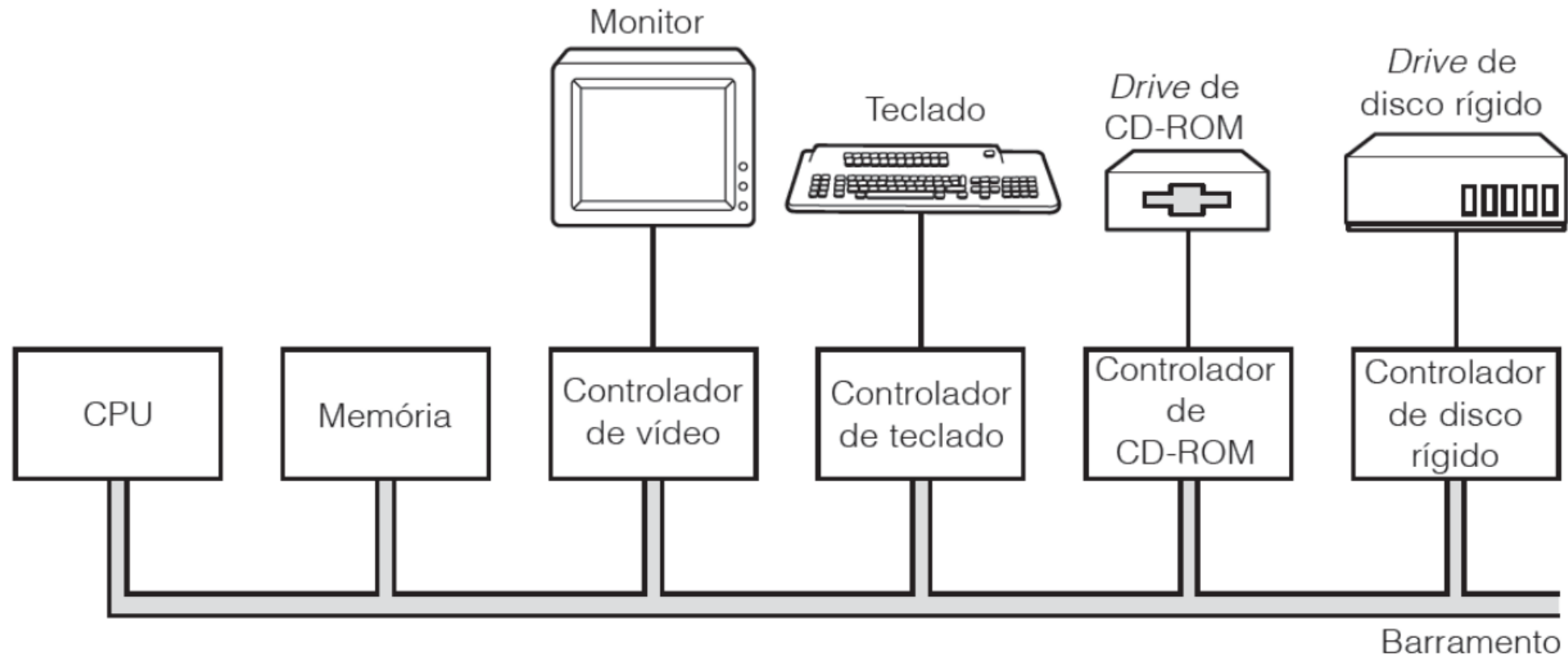
ENTRADA/SAÍDA

- A maioria dos computadores pessoais e estações de trabalho tem uma estrutura semelhante à mostrada abaixo:



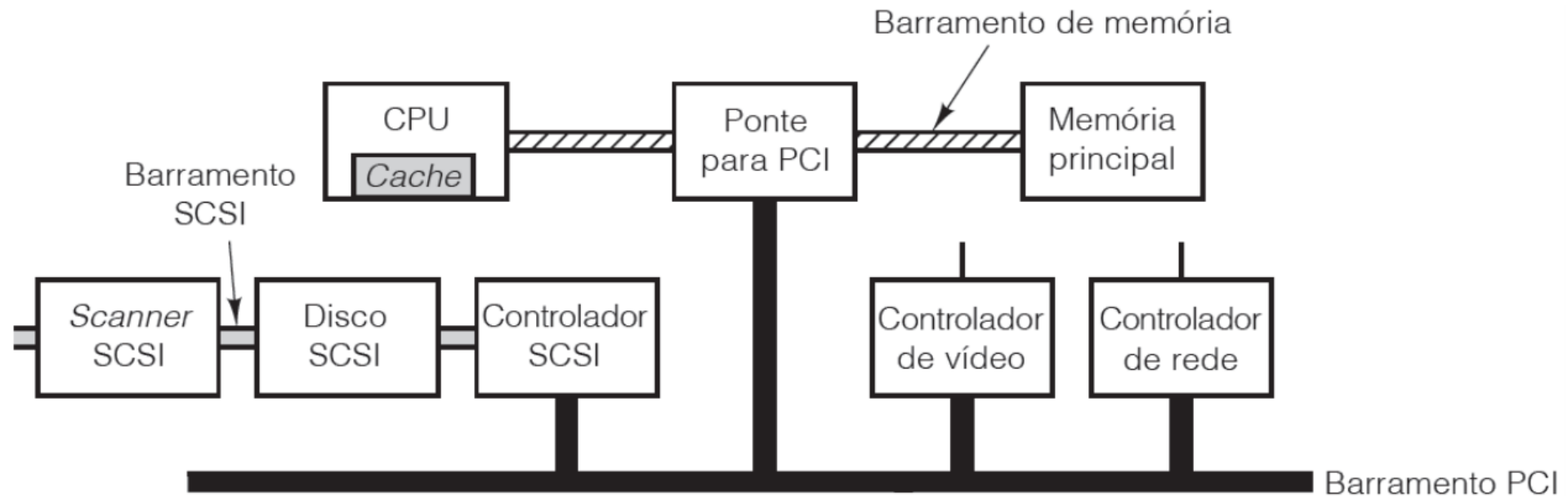
ENTRADA/SAÍDA

- A estrutura lógica de um computador pessoal simples pode ser vista abaixo:



ENTRADA/SAÍDA

- A função de um controlador é controlar seu dispositivo de E/S e manipular para ele o acesso ao barramento
- O mais popular deles é o **barramento PCI** - pode ser usado em muitas configurações, mas a figura abaixo ilustra uma configuração típica



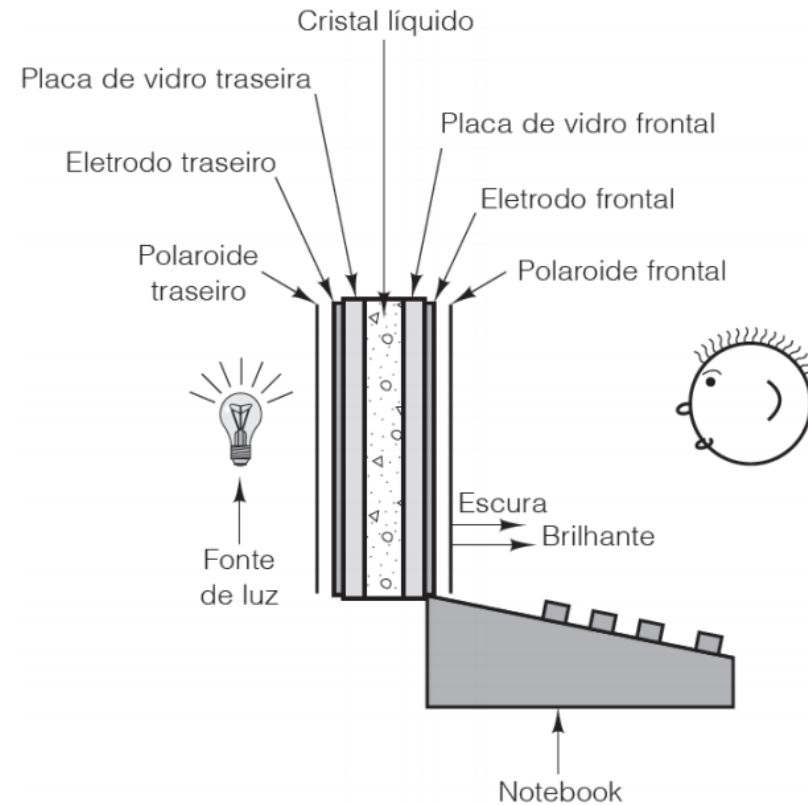
ENTRADA/SAÍDA

- Há muitos tipos de dispositivos de E/S disponíveis.
- Terminais de computador consistem em duas partes: um teclado e um monitor
- Dispositivos de toque (*touch screens*) podem ser encontrados em duas categorias: opacos e transparentes
 - Um dispositivo sensível ao toque opaco é o touchpad de um notebook
 - Um dispositivo transparente típico é a tela de um smartphone ou tablet



ENTRADA/SAÍDA

- A mais comum tecnologia de monitor de tela plana é o LCD
- Construção de uma tela de LCD:



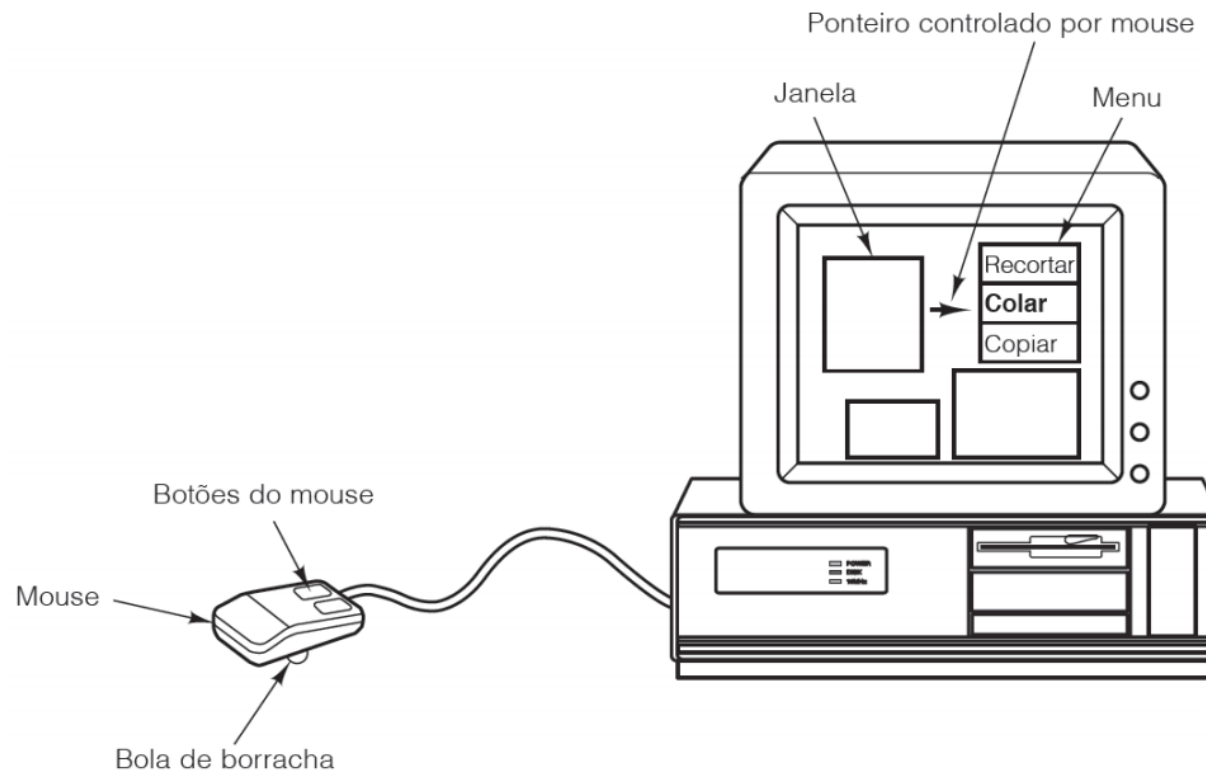
ENTRADA/SAÍDA

- Quase todos os monitores são renovados de 60 a 100 vezes por segundo por uma memória especial, denominada RAM de vídeo
- Essa memória tem um ou mais mapas de bits que representam a imagem da tela
- Em uma tela com, por exemplo, 1.920×1.080 elementos de imagem, denominados pixels, uma RAM de vídeo conteria 1.920×1.080 valores, um para cada pixel
- Ela poderia conter muitos desses mapas de bits, para permitir a passagem rápida de uma imagem para outra



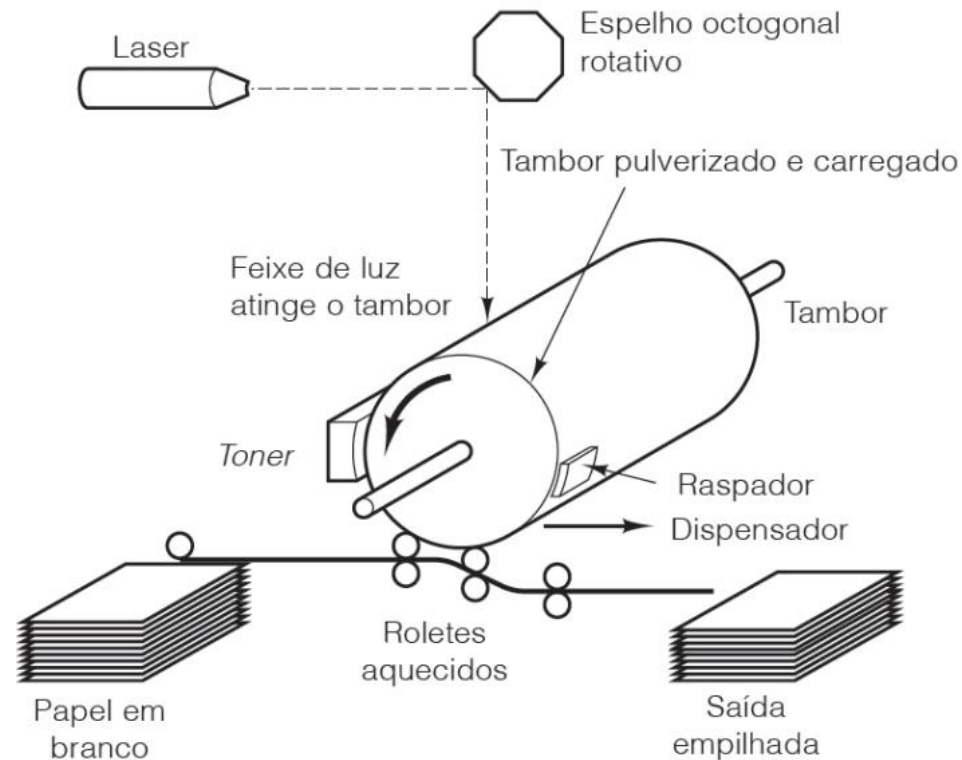
ENTRADA/SAÍDA

- O meio mais comum de permitir que usuários apontem algo na tela é um mouse



ENTRADA/SAÍDA

- Talvez o desenvolvimento mais interessante da impressão desde que Johann Gutenberg inventou o tipo móvel no século XV é a **impressora a laser**



ENTRADA/SAÍDA

- Para impressão doméstica de baixo custo, as **impressoras a jato de tinta** são as favoritas
- A cabeça de impressão móvel, que mantém os cartuchos de tinta, é varrida horizontalmente pelo papel por uma correia, enquanto a tinta é espirrada por minúsculos esguichos
- As gotículas de tinta têm um volume de mais ou menos 1 picolitro, de modo que 100 milhões delas formam uma única gota d'água



ENTRADA/SAÍDA

- Ainda outro tipo de impressora em cores é a **impressora por sublimação de corante**, ou de tinta
- Uma base contendo os corantes CMYK passa sobre um cabeçote de impressão térmico que contém milhares de elementos de aquecimento programáveis
- As tintas são vaporizadas instantaneamente e absorvidas por um papel especial que está próximo



ENTRADA/SAÍDA

- Uma utilização cada vez mais popular de computadores é a fotografia digital

