

Memória

- Nos últimos anos vem se investindo no aumento da velocidade dos processadores
- Porém, a velocidade de processamento de um sistema não é determinada somente pelo seu processador
- Não adianta ter o processador mais rápido do mundo se a alimentação de informações não consegue acompanhar o mesmo ritmo
- Como tanto o fornecimento dos dados como seu armazenamento após o processamento são efetuados na memória, a velocidade média de acesso a memória é importante no cálculo de velocidade de um sistema.

- Além da velocidade, o tamanho da memória também é importante.
- O ideal seria:
 - Memória de tamanho ilimitado;
 - Memória com um tempo de acesso muito rápido.
- Entretanto, esses são objetivos contraditórios:
 - Por problemas tecnológicos, quanto maior a memória mais lenta será o seu tempo de acesso
- Solução:
 - Criar uma ilusão para o processador de forma que a memória pareça ilimitada e muito rápida.

Princípio da Localidade

■ Localidade Temporal

- Se um item é referenciado, ele tende a ser referenciado novamente dentro de um espaço de tempo curto.

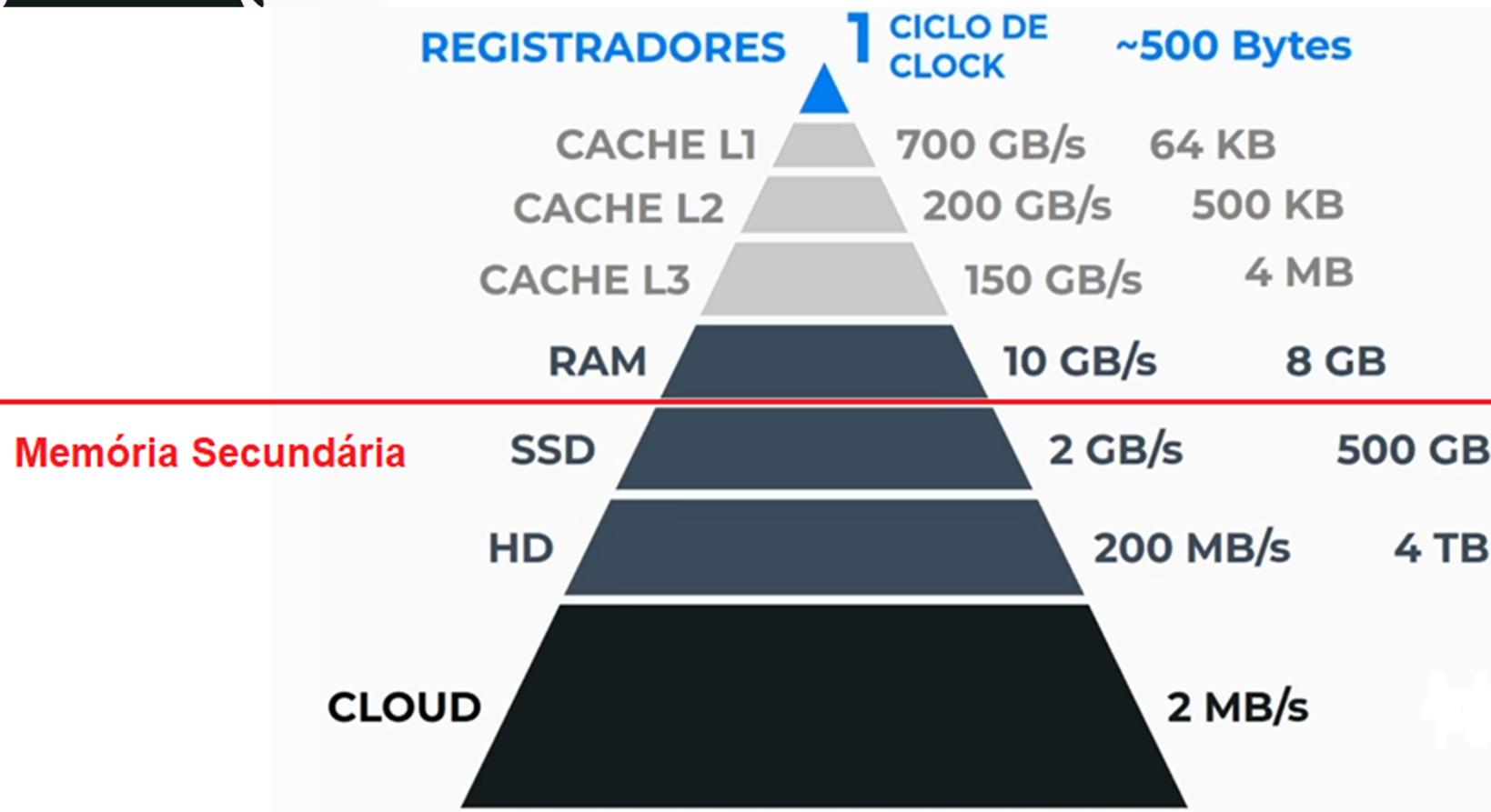
■ Localidade Espacial

- Se um item é referenciado, itens cujos endereços sejam próximos dele tendem a ser logo referenciados.

Hierarquia de Memória

- Ilusão de uma memória ilimitada e rápida obtida através da utilização de diversos níveis de acesso
- A hierarquia de memória explora o princípio da localidade
 - Localidade de memória é o princípio que diz que os próximos acessos ao espaço de endereçamento tendem a ser próximos

Hierarquia de Memória



Hierarquia de Memória

- A idéia de memória secundária já é aplicado a décadas.
- Os dados são transferidos para níveis mais altos a medida que são usados
- A transferência entre níveis é feita com grupos de palavra (bloco, página) pois o custo relativo de transferir um grupo de dados é menor do que para uma única palavra, além de já antecipar acessos (localidade espacial)

Hierarquia de Memória

- Vale lembrar que para movimentar dados entre os níveis são necessários mecanismos baseados em políticas
 - Ex: é preciso mover dados de um nível superior que já esta cheio. Alguém deve ser retirado? Quem?
 - Uma decisão errada pode afetar todo o desempenho do sistema

Hierarquia de Memória

Algumas métricas de medidas:

- Hit - dado encontrado no nível procurado.
- Miss - dado não encontrado no nível procurado.
- Hit-rate (ratio) - percentual de hits no nível.
- Miss-rate (ratio) – percentual de misses no nível. É complementar ao hit-rate.
- Hit-time – tempo de acesso ao nível incluindo tempo de ver se é hit ou miss.
- Miss-penalty – tempo médio gasto para que o dado não encontrado no nível desejado seja transferido dos níveis mais baixos.

Hierarquia de Memória

Exemplo de cálculo:

- Calcule o tempo médio (tme) efetivo de acesso a uma memória cache considerando
 - Hit-ratio = 80%
 - Hit-time = 2 μ s
 - Miss-penalty = 10 μ s
- $$\begin{aligned} \text{Tme} &= \text{hit-time} * \text{Hit-ratio} + (1 - \text{Hit-ratio}) * \text{miss-penalty} \\ &= 2 \text{ us} * 0,80 + (1 - 0,80) * 10 \text{ us} \\ &= 1,6 \text{ us} + 2 \text{ us} \\ &= 3,6 \text{ us} \end{aligned}$$

Características de Memória

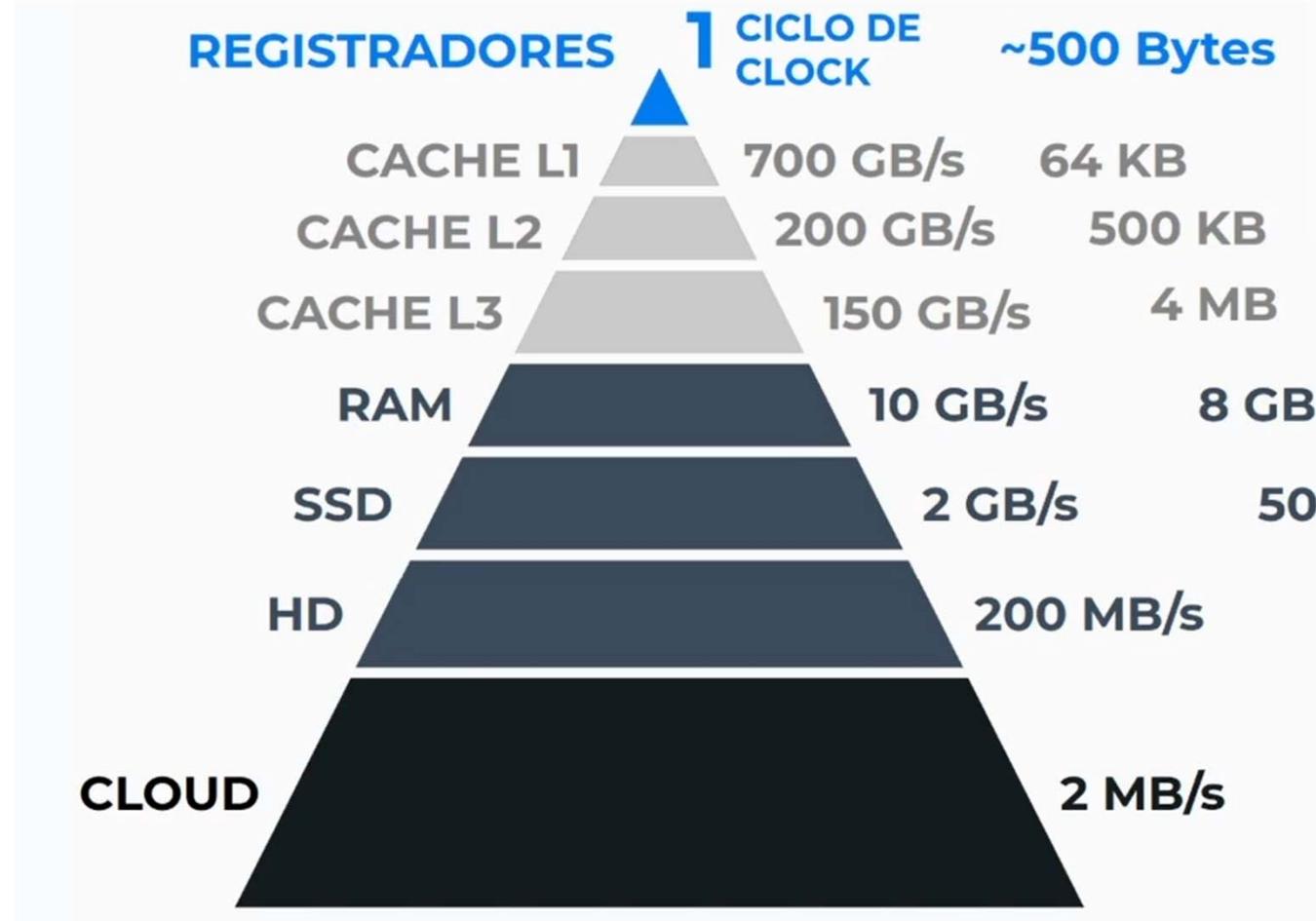
Memórias são estruturas físicas em forma de chips ou discos que são responsáveis pelo armazenamento de dados de forma temporária ou permanente que podem ser categorizadas como primárias ou secundárias.

As **memórias primárias** são aquelas em que o processador endereça diretamente como: RAM, ROM, Registradores.

Já as **memórias secundárias** são aquelas que não podem ser endereçadas diretamente, todas as informações deverão ser mandadas para uma memória intermediária antes (primária). Exemplos: HD, SSD, CD, DVD.

Podemos também entender melhor a tipologia, ou tipo de gravação, das memórias que podem ser **voláteis ou não-voláteis**.

Hierarquia de Memória



Hierarquia de Memória

Tipo	Tempo de acesso	Custo (por MB)
Registradores	Ciclos de CPU	---
Cache interna L1	Ciclos de CPU	---
Cache externa L2	8-35 ns	50 Us\$
Memória Principal	40-120 ns	1 Us\$
Memória secundária	5 ms	0,02 Us\$

Registradores

Os processadores possuem espaços específicos onde são guardados valores, os chamados registradores. Esses espaços são parecidos com variáveis de uma linguagem de programação de alto nível, onde se guarda um valor qualquer até que este seja modificado.

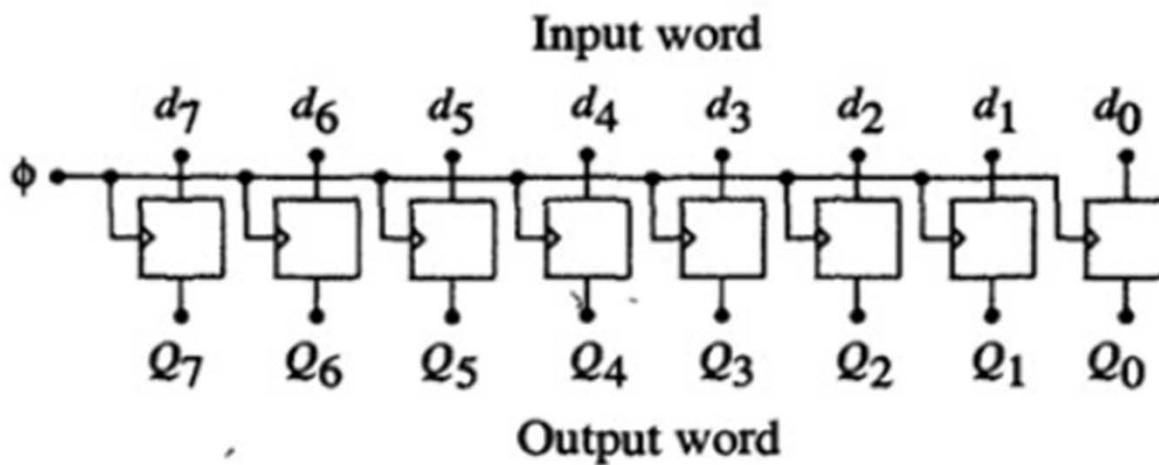
Os registradores estão no topo da hierarquia da memória e propiciam ao sistema o modo mais rápido de acessar dados. O termo geralmente é usado para designar apenas o grupo de registradores que podem ser utilizados diretamente para entradas ou saídas através de instruções definidas no conjunto de instruções do processador.

O conjunto de instruções da arquitetura x86, por exemplo, define um conjunto de 8 registradores de 32 bits, mas uma CPU deste tipo possui mais registradores do que apenas estes. Uma arquitetura MIPS 32 bits possui 32 registradores, dentre os 32, apenas alguns podem ser associados às variáveis, outros são utilizados para o controle da máquina, da pilha, argumentos de funções ou mesmo registradores temporários.

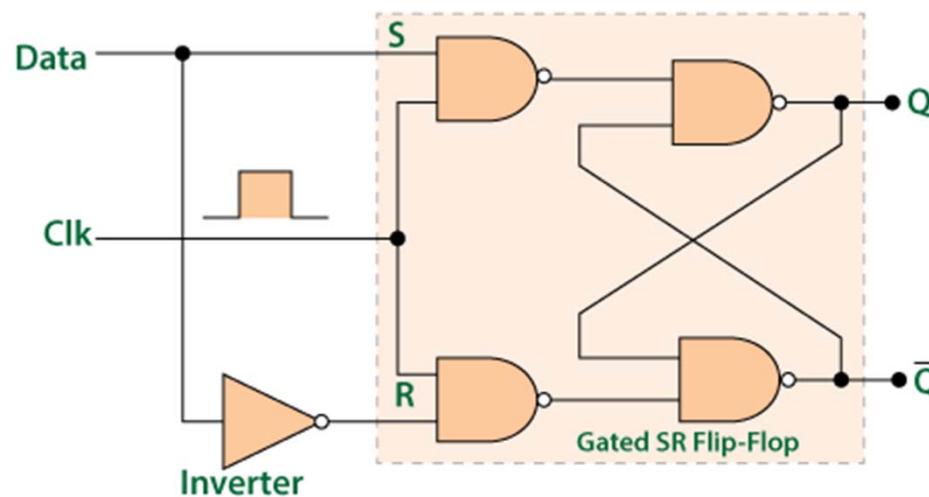
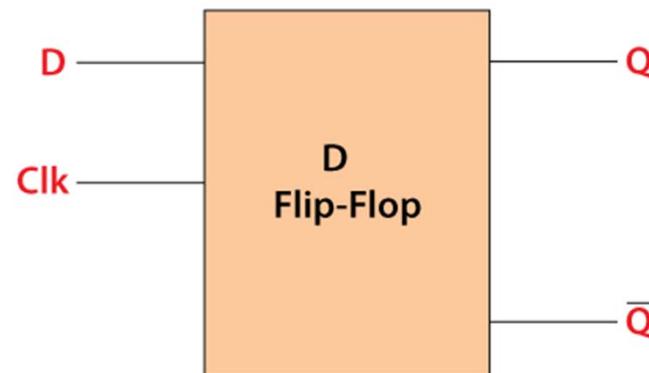
Os registradores são medidos pelo número de bits que podem conter. Atualmente as máquinas possuem registradores de 32 ou 64 bits. De acordo com o conteúdo, os registradores podem ser classificados em registradores de uso geral, de segmento, de ponteiro e de estado.

- Construídos a partir do agrupamento de flip-flops.
- Um registrador é um elemento lógico usado para armazenar uma palavra binária de n bits.

Registrador de 8 bits



O que é um Flip-Flop tipo D

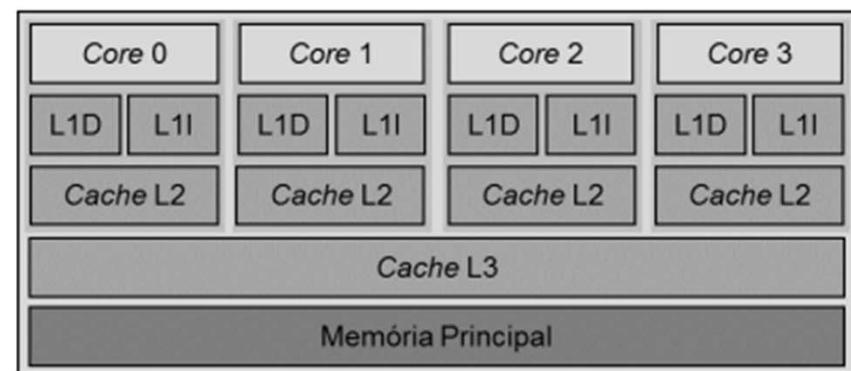
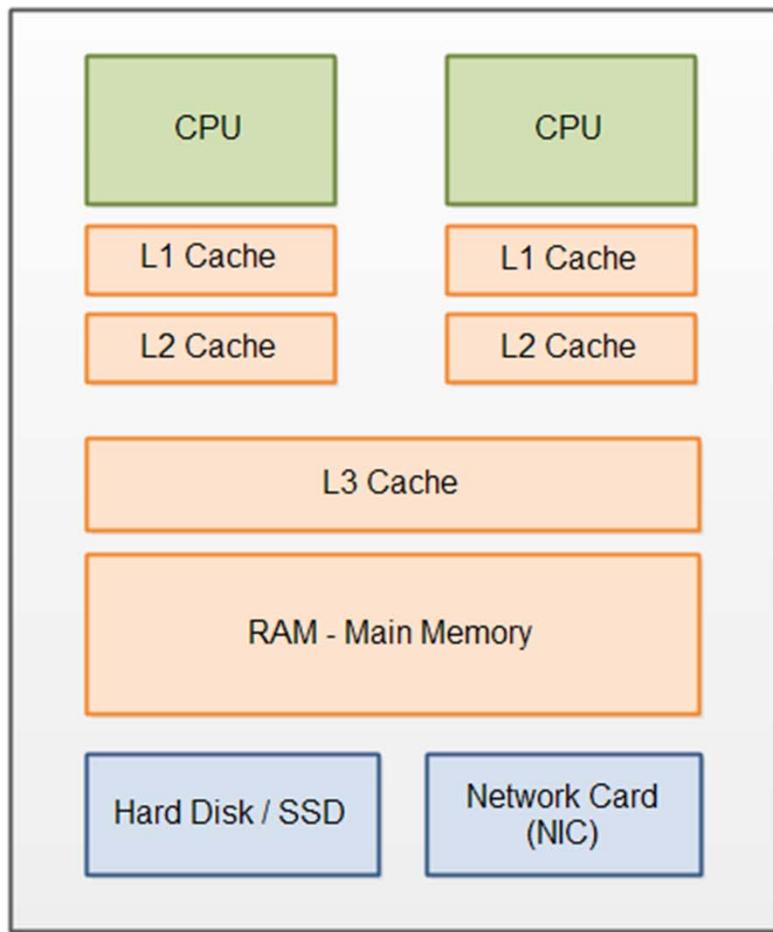


Memória cache

A memória cache é um recurso existente nos computadores que funciona em conjunto com o processador, ajudando a controlar e gerenciar o processamento de dados, facilitando o acesso a informação e reduzindo o tempo de execução dessa leitura.

Explicando de maneira mais simplista, a memória cache foi criada como uma memória complementar, com intuito de oferecer mais agilidade à performance da máquina, priorizando os processamentos cruciais para funcionamento do sistema.

Assim, ela funciona como um local de armazenamento temporário de dados, no qual é armazenado um bloco de informações trazidos da memória RAM, para que o processador possa consultar esse bloco com mais facilidade e rapidez, ao invés de ter que fazer a busca em toda a memória RAM, a cada nova consulta.



Memória RAM

É o dispositivo usado pelo processador para armazenar os arquivos e programas que estão sendo processados.

A quantidade de RAM disponível tem um grande efeito sobre o desempenho, já que sem uma quantidade suficiente dela o sistema passa a usar memória virtual, que é lenta.

A principal característica da RAM é que ela é volátil, ou seja, os dados se perdem ao reiniciar o computador. Ao ligar é necessário refazer todo o processo de carregamento, em que o sistema operacional e aplicativos usados são transferidos do HD para a memória, onde podem ser executados pelo processador.

Os chips de memória são vendidos na forma de pentes de memória. Existem pentes de várias capacidades e normalmente as placas mães possuem dois ou três encaixes disponíveis.

A RAM pode ser

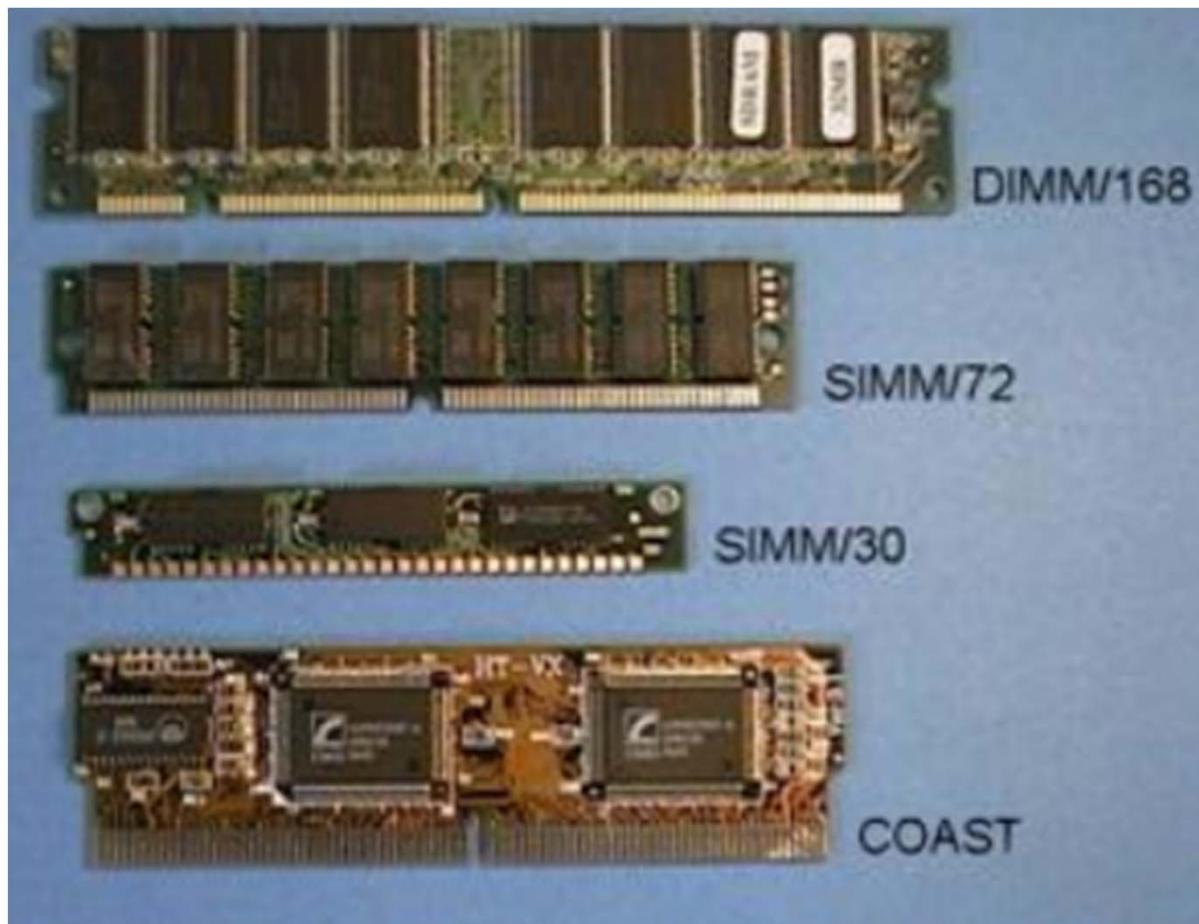
dinâmica — DRAM — onde existe uma necessidade de atualização dos dados **ou**

estática — SRAM — que guarda dados mais antigos uma vez que não necessita de *refresh*.

	DRAM (dinâmica)	SRAM (estática)
Vantagens	<ul style="list-style-type: none">- alta densidade de integração- baixo consumo de potência- baixa geração de calor- baixo custo	<ul style="list-style-type: none">- alta velocidade- não precisam de “refresh”
Desvantagens	<ul style="list-style-type: none">- baixa velocidade- necessidade de refresco (“refresh”)	<ul style="list-style-type: none">- baixa densidade de integração- alto consumo de potência- alta geração de calor- alto custo
Tempo de Acesso	60 a 70 ns	10 a 20 ns

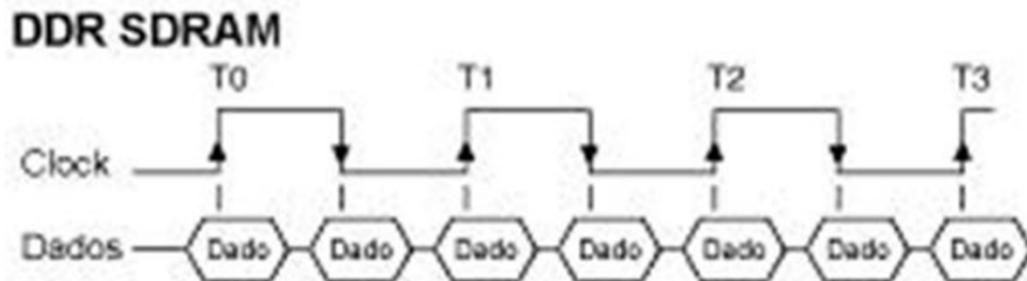
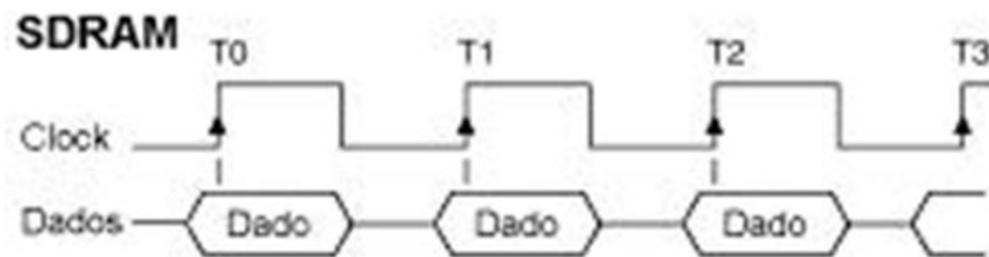
Tipos: SIMM e DIMM

Os tipos podem ser diferenciadas pela conexão com o soquete, podendo ser SIMM (Single InLine Memory Module) ou DIMM (Double InLine Memory Module).



Focando nas memórias DIMM, elas podem ser:

- **SDRAM** onde os ciclos da memória são sincronizados com os ciclos da placa-mãe, permitindo um maior desempenho (O S é de Síncrono);
- **DDR-SDRAM** que fornece o dobro da informação no mesmo intervalo de tempo.



DDR, DDR2, DDR3, DDR4, etc...

DDR2 e DDR

A diferença está principalmente nos MHZ, enquanto a primeira possui 266, 333 e 400MHZ, a segunda tem valores mais expressivos: 400, 533, 667 e 800MHZ.

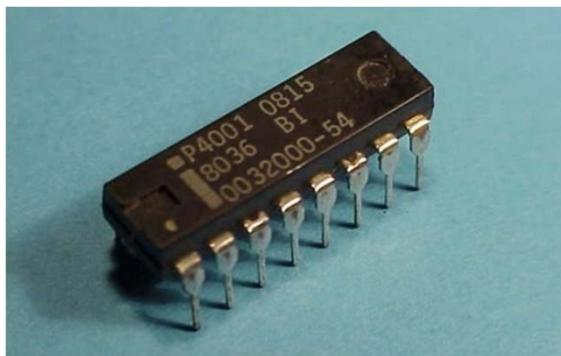
Memórias DDR3–1333 trabalham externamente a 666,6 MHz, memórias DDR4–2133 trabalham externamente a 1.067 MHz, e assim por diante.

Ambas transmitem dois dados por ciclo de *clock*, logo a frequência será a metade.

Memória ROM

A memória ROM (Read Only Memory) **permite apenas a leitura de dados**, a escrita é feita por processo especial durante a fabricação do chip.

A ROM é consideravelmente mais barata que a RAM. Independentemente de se desligar o computador os dados não são perdidos.



ROM



EPROM



EEPROM em uma placa mãe

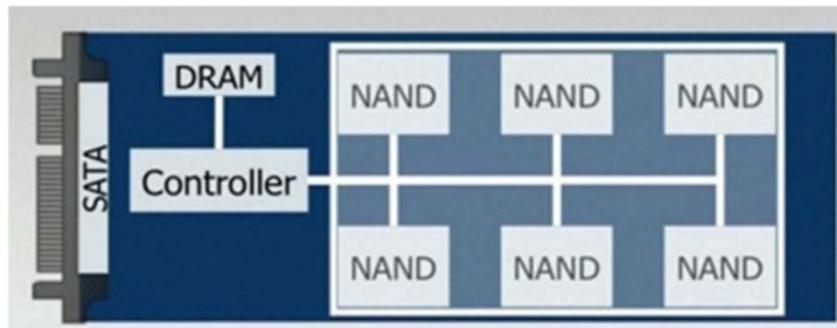
- **PROM** (Programmable ROM) — Gravação feita após a fabricação através da queima dos fusíveis sendo feita somente uma vez;
- **EPROM** (Erasable PROM) — Pode ser programada e apagada várias vezes, semelhante à RAM estática. No entanto, para apagar o seu conteúdo, é necessário expor a memória a uma luz ultravioleta;
- **EEPROM** (Electrically EPROM) — A programação, o apagamento e a reprogramação são feitas por controle do processador. São lentas e caras em relação às RAMs e ROMs, tendo baixa velocidade e capacidade de armazenamento;
- **Memória Flash** (Tipo especial de EEPROM) — Pode ser apagada e regravada em blocos e só suporta reprogramações.

Tipo	Categoria	Forma de apagar
SRAM	L/E	Eletricidade
DRAM	L/E	Eletricidade
ROM	L	Não é possível
PROM	L	Não é possível
EPROM	quase sempre L	Luz ultravioleta
EEPROM	quase sempre L	Eletricidade
Flash	L/E	Eletricidade

SSD

SSD ou Solid State Drive é um tipo de memória não volátil (similar à memória Flash) e é considerada a evolução dos hard disks.

Esses semicondutores possuem controladores e memórias flash NAND, que utilizam corrente elétrica para o armazenamento de dados, conforme a figura.



Diferente dos HDDs, o Solid State Drive não possui partes mecânicas como discos magnéticos, motores, braços e cabeçote de gravação.

Justamente por não possuírem partes móveis e serem classificadas como a próxima geração de hard drives, bem como dispositivos_de_armazenamento, as memórias SSD foram batizadas como "hard disks de estado sólido".

Ainda na Figura , temos representado à esquerda uma interface de fluxo de dados do tipo Serial ATA, podendo esta atingir taxas de transferência de até 6 gigabits por segundo.

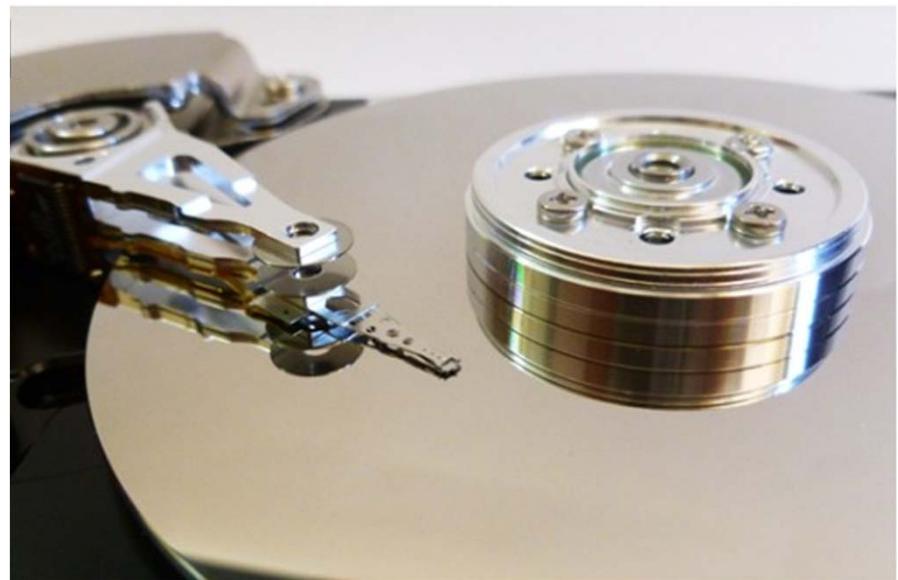
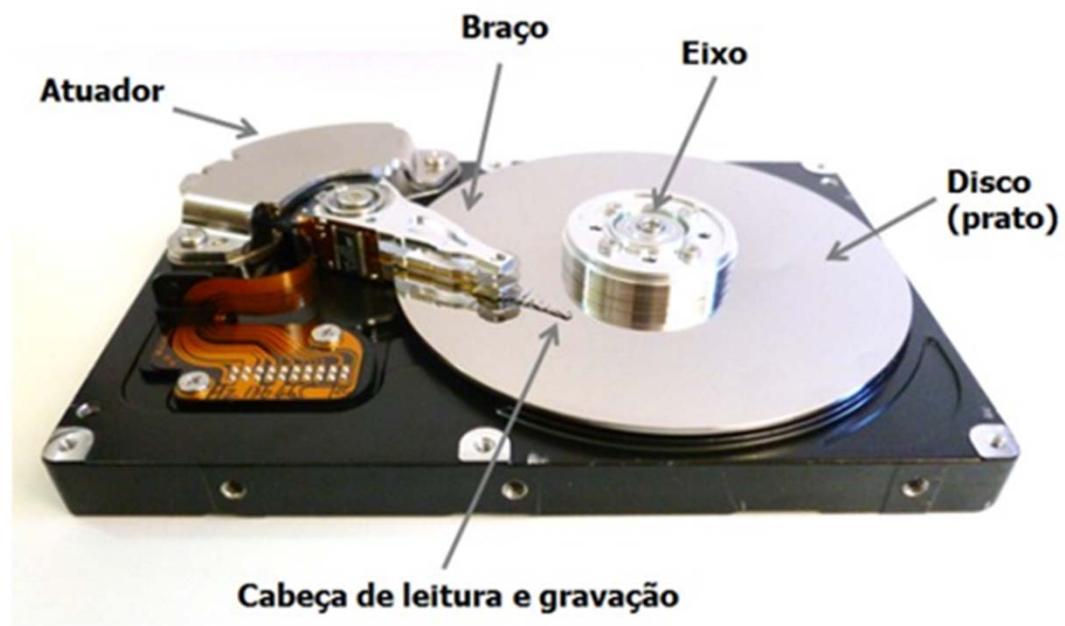
HD

O **disco rígido** — **HD** (*Hard Disk*) ou **HDD** (*Hard Disk Drive*) — é o dispositivo de armazenamento de dados mais utilizado nos computadores. Trata-se de um dispositivo mecânico baseado no efeito magnético.

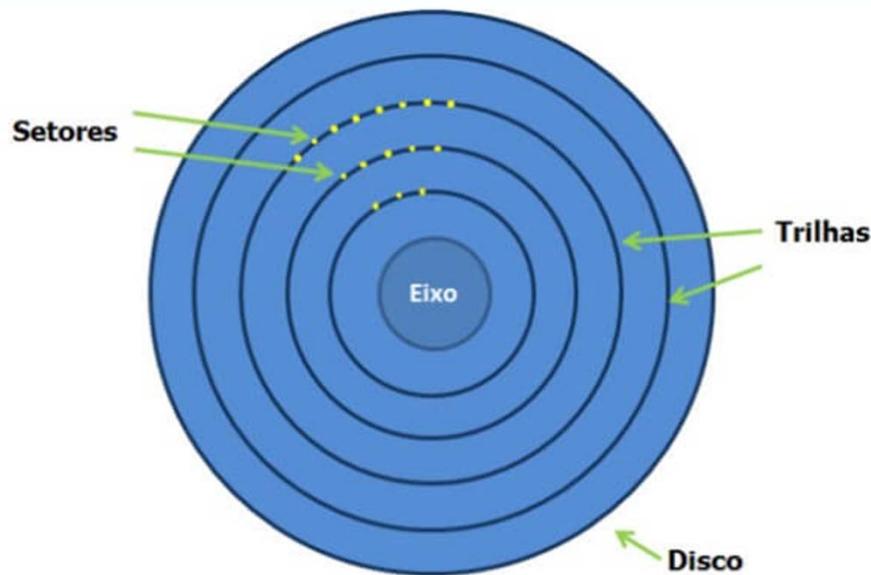
Internamente os **pratos** correspondem aos discos nos quais os dados são efetivamente armazenados. Eles são feitos, geralmente, de alumínio ou de um tipo de cristal recoberto por material magnético e por uma camada de material protetor.

Quanto mais denso for o material magnético, maior é a capacidade de armazenamento do disco. Note que os HDs com grande capacidade contam com mais de um prato, um sobre o outro. Eles ficam posicionados em um **eixo** responsável por fazê-los girar.

Os HDs mais comuns giram a 7.200 RPM (rotações por minuto), mas também há modelos que alcançam a taxa de 10.000 ou até 15.000 rotações. Em notebooks, convencionou-se o uso de discos rígidos com 5.400 RPM.



Para a "ordenação" dos dados no HD, é utilizado um esquema conhecido como **geometria dos discos**. Nele, o disco é dividido em *cilindros*, *trilhas* e *setores*:



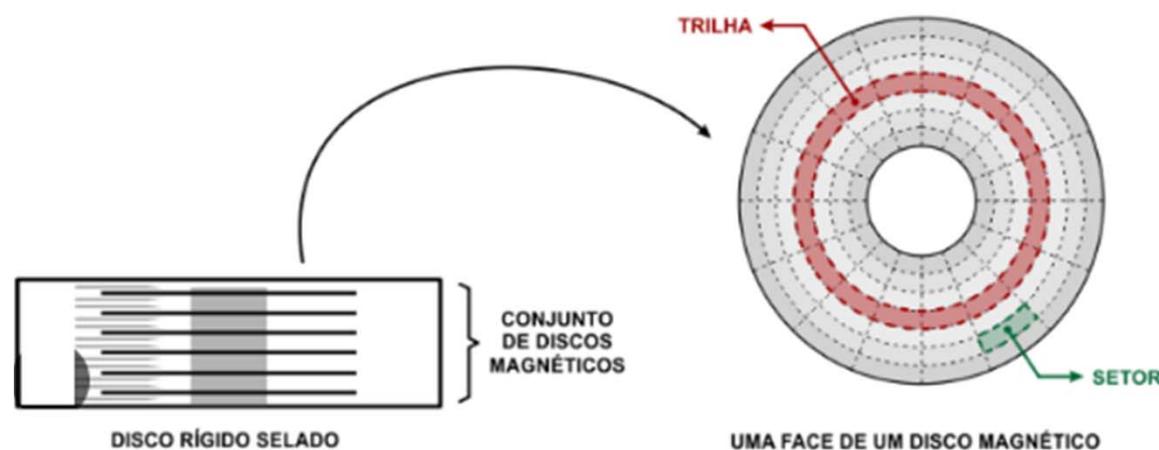
As *trilhas* são círculos que começam no centro do disco e vão até a sua borda, como se estivessem um dentro do outro. Essas trilhas são numeradas da borda para o centro: a trilha que fica mais próxima da extremidade do disco é denominada *trilha 0*, a trilha que vem em seguida é chamada de *trilha 1* e assim por diante, até chegar à trilha mais próxima do centro.

Cada trilha é dividida em trechos regulares denominados *setores*. Cada setor possui uma capacidade específica de armazenamento (geralmente, de 512 bytes).

E os *cilindros*? Eis um ponto interessante. Você já sabe que um HD pode conter vários pratos, sendo que há uma cabeça de leitura e gravação para cada lado dos discos. Imagine que é necessário ler a trilha 42 do lado superior do disco 1. O braço movimentará a cabeça até essa trilha, mas fará as demais se posicionarem de forma igual.

Isso ocorre porque, normalmente, o braço se movimenta de uma só vez, isto é, ele não é capaz de mover uma cabeça para uma trilha e uma segunda cabeça para outra.

Assim, quando a cabeça é direcionada à trilha 42 do lado superior do disco 1, todas as demais cabeças ficam posicionadas sobre a mesma trilha, só que em seus respectivos discos. Quando isso ocorre, damos o nome de *cilindro*. Em outras palavras, cilindro é a posição das cabeças sobre as mesmas trilhas de seus respectivos discos.



Dentro do disco rígido, os discos magnéticos giram continuamente. Por isso, dificilmente os setores a serem lidos estarão sob a cabeça de leitura/gravação no exato momento de executar a operação. No pior dos casos, pode ser necessária uma volta completa do disco até o setor desejado passar novamente sob a cabeça de leitura.

O tempo de latência é tão importante quanto o tempo de busca. Felizmente, ele é fácil de ser calculado, bastando dividir 60 pela velocidade de rotação do HD em RPM (rotações por minuto), e multiplicar o resultado por 1000. Teremos então o tempo de latência em milissegundos. Um HD de 5400 RPM, por exemplo, terá um tempo de latência de 11.11 milissegundos (o tempo de uma rotação), já que $60 \div 5200 \times 1000 = 11.11$.

Geralmente é usado o tempo médio de latência, que corresponde à metade de uma rotação do disco (assumindo que os setores desejados estarão, em média, a meio caminho da cabeça de leitura). Um HD de 5400 RPM teria um tempo de latência médio de 5.55 ms, um de 7.200 RPM de 4.15 ms e um de 10.000 RPM de apenas 3 ms.

Muitos fabricantes publicam o tempo de latência médio nas especificações ao invés do tempo de busca ou o tempo de acesso (já que ele é menor), o que acaba confundindo os desavisados.

Custo de acesso a disco

Alguns conceitos importantes:

- **Latência rotacional:** é o tempo gasto para localizar o setor ao qual se quer ter acesso.
- **Tempo de seek:** é o tempo gasto para a cabeça de leitura/gravação se posicionar na trilha correta.
- **Tempo de transferência:** é o tempo gasto para a migração dos dados da memória secundária para a memória principal.
- **Tempo de acesso:** seek + latência + transferência

Memória secundária - **Custo de acesso a disco**

custo com acesso seqüencial

- Custo = tempo para leitura do arquivo, setor por setor, em seqüência
- Para cada trilha (seqüência de setores consecutivos), o processo de leitura envolve as seguintes medidas:
 - tempo médio de seek: 18 msec
 - latência rotacional: 8.3 msec
 - tempo de transferência (leitura de uma trilha) : 16.7 msec
- Custo de acesso a uma trilha= $18 + 8.3 + 16.7 = 43$ msec
- Custo de acesso ao arquivo:
 - Para 100 trilhas: 100×43 msec = 4.300 msec = **4.3 sec**

Detecção de erro

Exemplo: bit de paridade

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Manter a qtd. de um's
par.

Definições

Distância de Hamming (ou distância de erro)

é o número de posições (bits) em que as palavras diferem.

Exemplo:

$$\begin{array}{r} p_1 = 1 \ 0 \ 0 \boxed{0} \boxed{0} \boxed{1} \ 0 \ 0 \ 1 \\ p_2 = 1 \ 0 \ 1 \boxed{1} \boxed{1} \ 0 \ 0 \ 0 \ 1 \end{array} \left. \right\} \rightarrow \text{D. Hamming} = 3$$

$$xor(p_1, p_2) = 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0$$

Observação importante => Detecção de erros de 1 bit

**Bit de paridade detecta erros em palavras à distância
Hamming de 1.**

Seja 1111 uma palavra correta.

**Então : 1110, 1101, 1011 e 0111 são palavras com erros, pois a
Qtd. de um's não é par.**

- 1) Essas palavras erradas estão à distância de erro de 1.**
- 2) Palavras à distância Hamm. de 2 estão corretas.**

Definições

Palavra de código = palavra total a ser transmitida = n

Palavra válida = palavra que faz parte do vocabulário da máquina = m

Bits verificadores = bits incorporados à palavra válida para permitir a detecção ou correção do erro = r

Exemplo: seja uma palavra válida=100, quando adicionamos o bit verificador e usarmos a paridade, teremos uma palavra de código:

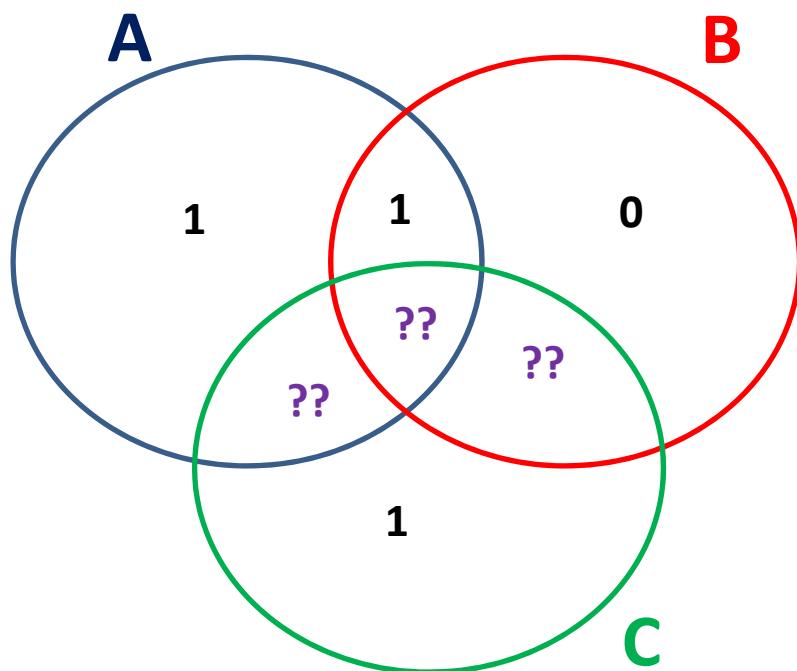
$$m = 100, r = 1, n = 1001$$

Regra: $n = m + r$

Onde n , m , r representam o número de bits

O Código de Hamming para correção de erros simples de 1 bit

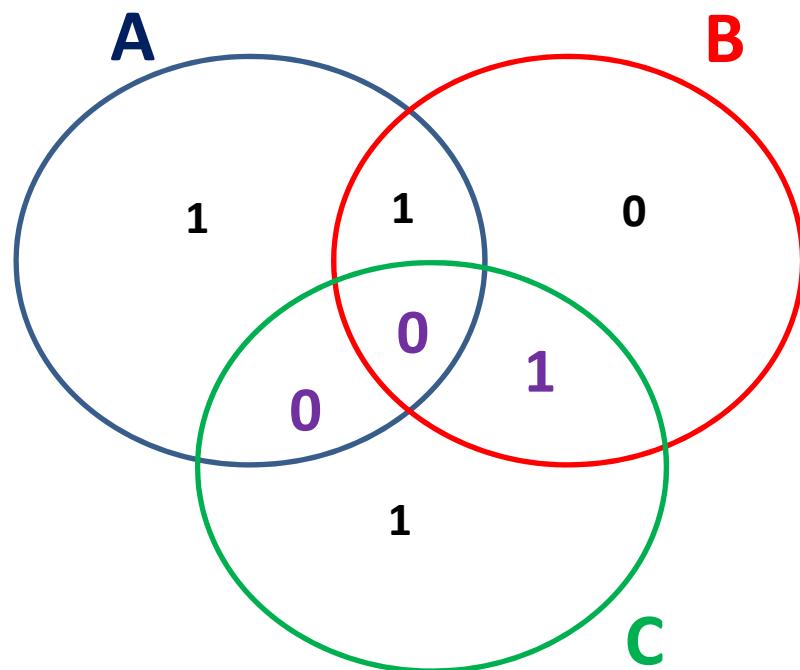
Seja uma palavra válida = 1101 colocada no diagrama de Venn a seguir:



Determinar quais os valores de cada uma das áreas dos círculos para que tenhamos uma paridade par em cada círculo, A, B e C.

O Código de Hamming para correção de erros simples de 1 bit

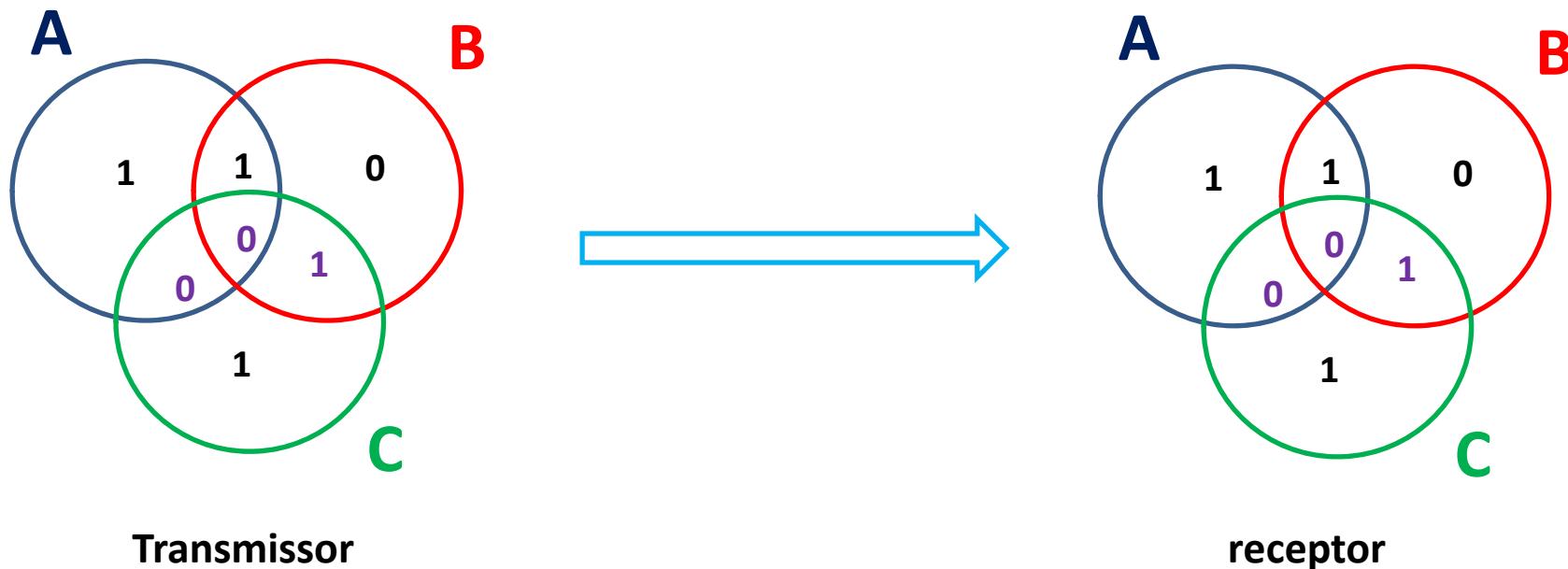
Seja uma palavra válida = 1101 colocada no diagrama de Venn a seguir:



Uma possível solução

O Código de Hamming para correção de erros simples de 1 bit

Agora vamos supor uma transmissão onde um erro simples de 1 bit irá acontecer



O receptor procurará preservar a paridade e a única forma será ajustar o bit que foi trocado durante a transmissão, esse bit, mesmo desconhecido, será encontrado e corrigido.

- Pelo que vimos, caso tenhamos 4 bits para as palavras válidas (ou m) e 3 bits para a verificação (ou r), o sistema funciona, isto é, conseguimos descobrir qual o bit está trocado.
- Bem, sabemos que $n = m + r$, assim temos:
 $n = 7$ bits para o total de palavras (corretas ou não)
 $m=4$ bits para as palavras válidas
 $r = 3$ bits para a verificação

- Temos que responder agora duas perguntas:
 1. Quantos bits devo adicionar a uma palavra válida;
 2. Onde colocar estes bits

Quantos bits devo adicionar a uma palavra válida?

Do nosso exemplo inicial temos:

1101 001

Palavra de código
Correta => $n = 7$
 $m = 4$
 $r=3$
 $n = m + r$



1101000
1101011
1101101
1100001
1111001
1001001
0101001

A partir da palavra de código original, Podemos ter 7 palavras erradas. Estas 7 palavras correspondem a um bit trocado de cada vez na palavra de código correta.

- Resumindo:

Cada palavra de código correta pode gerar até 7 palavras de código erradas.

Como tenho apenas m bits para as palavras válidas, terei um total de 2^m palavras .

Mas tenho n bits para as palavras de código, o que me dará um total de 2^n palavras.

Como cada palavra válida pode gerar até 7 (ou n) palavras erradas temos:

$$2^n \geq 2^m \cdot (1 + n)$$

Mas, $n = m + r$

Assim, $2^n \geq 2^m \cdot (1 + n)$ pode ser escrito como:

$$2^{m+r} \geq 2^m \cdot (1 + m + r) \text{ ou}$$

$$2^m \cdot 2^r \geq 2^m \cdot (1 + m + r) \text{ ou } 2^r \geq 1 + m + r$$

Esta é a regra que irá determinar a quantidade de bits de verificação (r) que deveremos anexar à palavra válida m e termos a palavra de código n .

Exemplo:

 $m=4 \Rightarrow$ Dual de valor de x \Rightarrow $2^2 \geq 1+4+2 \quad F$
 $3 \Rightarrow 2^3 \geq 1+4+3 \quad OK$

$m=5$, Dual valor de r \Rightarrow $2^2 \geq 1+5+2 \quad F$

$3 \Rightarrow 2^3 \geq 1+5+3 \quad F$

$4 \Rightarrow 2^4 \geq 1+5+4 \quad OK$

Mas, se $M=5 \Rightarrow r=4$.

Mas se $M=6 \Rightarrow r=4$

⋮

$M=11 \Rightarrow r=4$ ou $2^4 \geq 1+11+4 \quad OK$

Ou seja, com 4 bits de reescrita, a palavra
Válida poderá ter 11 bits! ?
 \Rightarrow porque $0 \geq e$ não é na fórmula!

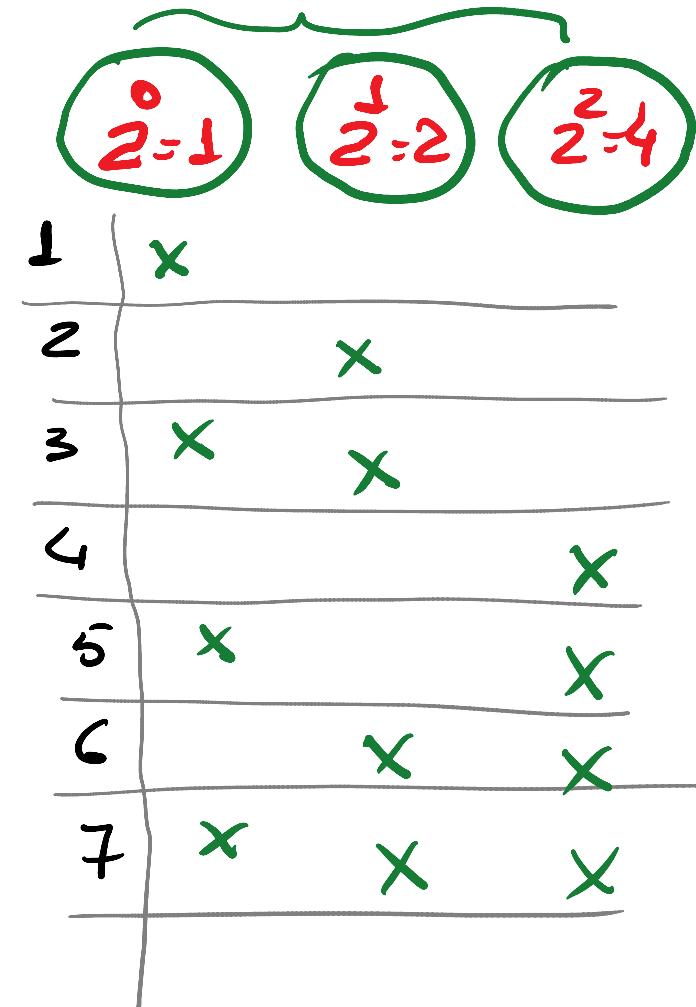
2) Onde colocar os bits?

Exemplo: palavra válida = 1101 (ou 13₍₁₀₎)

Como $m=4 \Rightarrow r=3$

1	0	1	0	1	0	1
1	2	3	4	5	6	7

$$\begin{aligned}2^0 &= 1, 3, 5, 7 & \Rightarrow \text{palavra de código} = \\2^1 &= 2, 4, 6, 7 & 85_{(10)} \\2^2 &= 4, 5, 6, 7\end{aligned}$$



Vamos supor que ao transmitirmos a palavra de código 85, um erro ocorra e recebamos por exemplo 81 ! Aconteceu 1 erro simples, mas quem está errado?

$\begin{array}{ c c c c c c c } \hline 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ \hline \end{array}$						
1	1	0	1			
1.	3.	5.	7	<u>erro</u>		
0	1	0	1			
2.	3.	6.	7	OK		
0	0	0	1			
4.	5.	6.	7	<u>erro</u>		

único bit que corrige todas as mesmas tempos é o bit 5, assim apólaora recebida foi: 1010101

$$1 + 4 = \textcircled{5}$$

$$= 85_{(10)}$$

Exercícios para método de Hamming para detecção e correção de erros simples de 1 bit:

- 1) Considere uma máquina onde as palavras válidas possuam 11 bits. Quero transmitir a palavra válida $601_{(10)}$. Qual a palavra de código gerada?**

- 2) Considere uma máquina onde as palavras de código possuam 12 bits.**
Recebi a palavra de código $3239_{(10)}$.
Qual a palavra válida transmitida?
Em qual bit o erro ocorreu?