

Tipos 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.

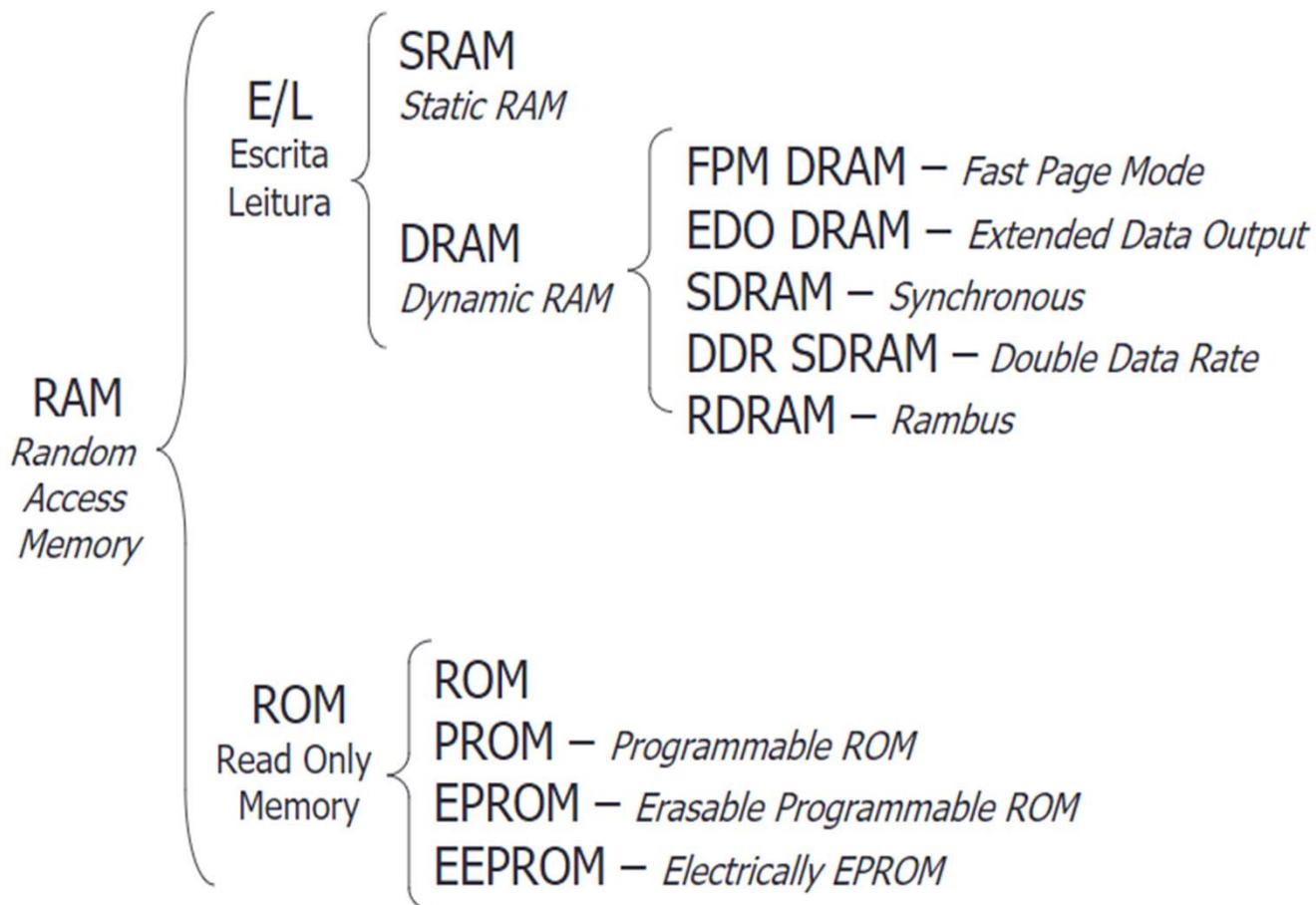
Já as **memórias secundárias** são aquelas que não podem ser endereçadas diretamente, todas as informações deverão ser mandadas para uma memória intermediária antes (primária). Exemplos: HD, SSD, CD, DVD.

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

Hierarquia de memória



Memória RAM



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

FPM DRAM (Fast Page Mode DRAM ou Memória DRAM de Modo de Operação Rápida)

O FPM já foi um dos mais usados entre os métodos de acesso de DRAM. Hoje só é usado em sistemas antigos. O ponto positivo é o baixo consumo, pois não é preciso correntes de sinal e recuperação durante os acessos de página. Existem pontos negativos, o mais relevante é que os buffers de saída desligam quando o sinal CAS sobe. Para que os buffers sejam desligados demora no mínimo 5ns, o que faz com que o ciclo fique 5ns mais lento.

As memórias FPM foram utilizadas em micros 386, 486 e nos primeiros micros Pentium, na forma de módulos SIMM de 30 ou 72 vias, com tempos de acesso de 80, 70 ou 60 ns, sendo as de 70 ns as mais comuns.

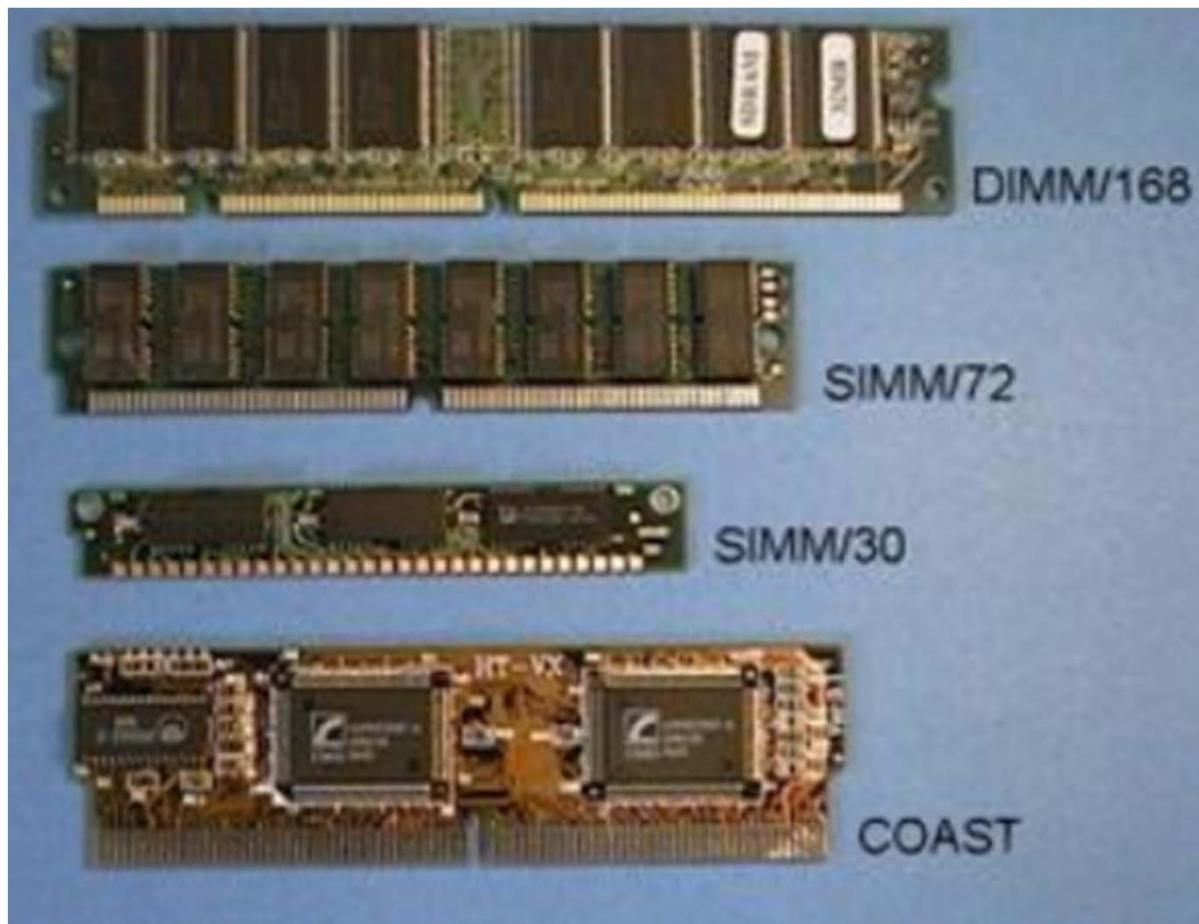
EDO DRAM (Extended Data Out DRAM ou Saída de Dados Estendida) é também conhecida como Hyper-page Mode DRAM.

A EDO foi um grande avanço em relação à FPM e funciona cerca de 40% mais rápido e tem o custo de fabricação praticamente o mesmo. Ela funciona bem em clocks de 83MHz, velocidade em que é equivalente a uma SDRAM. Em clocks de até 100MHz com chips de 55ns ou mais rápidos funciona também, porém com desempenho muito inferior a uma SDRAM.

As memórias EDO foram utilizadas, assim como as FPM, em micros Pentium na forma de módulos SIMM, mas com tempos de acesso menores. [Hoje a EDO é praticamente obsoleta, sendo pouco produzida, aos poucos será totalmente substituída pela SDRAM.](#)

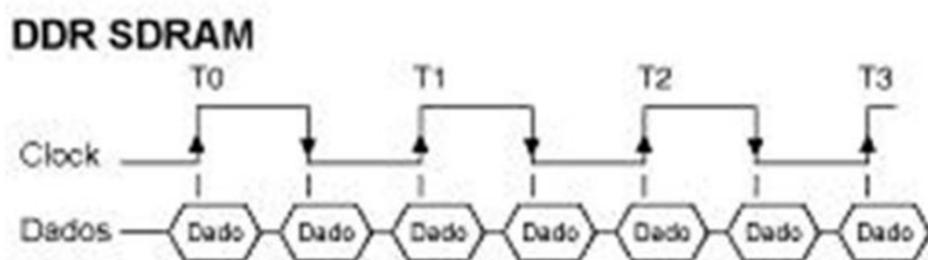
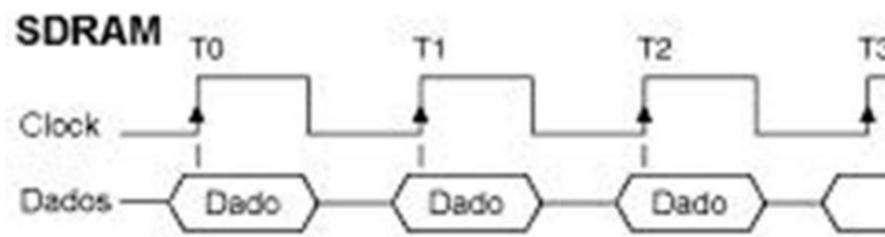
Tipos: SIMM e DIMM

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.

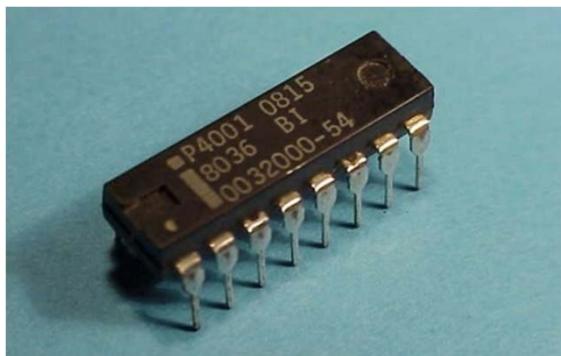
Memórias DDR3–1333 trabalham externamente a 666,6 MHz, memórias DDR4–2133 trabalham externamente a 1.067 MHz, e assim por diante.

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

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:

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

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

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:

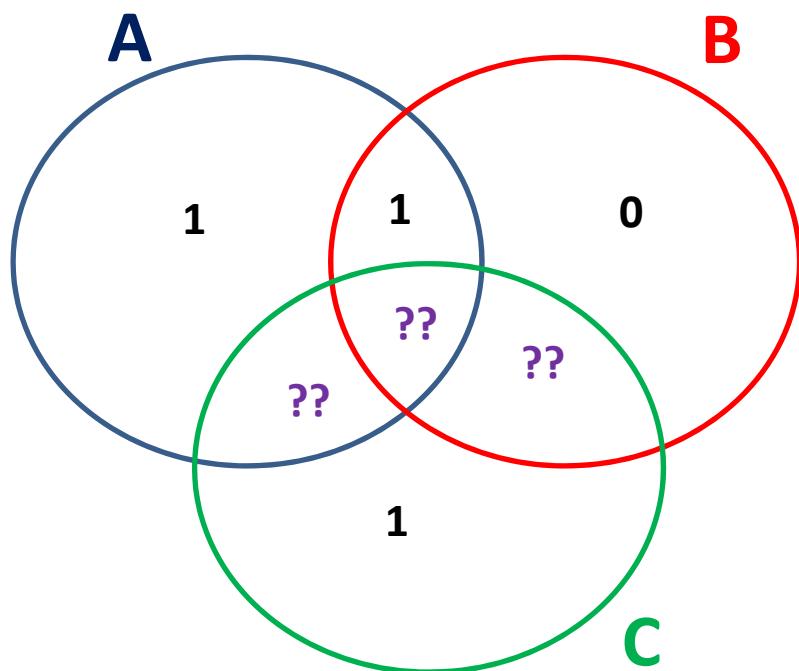
$$m = 100, r = 1, n = 1001$$

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

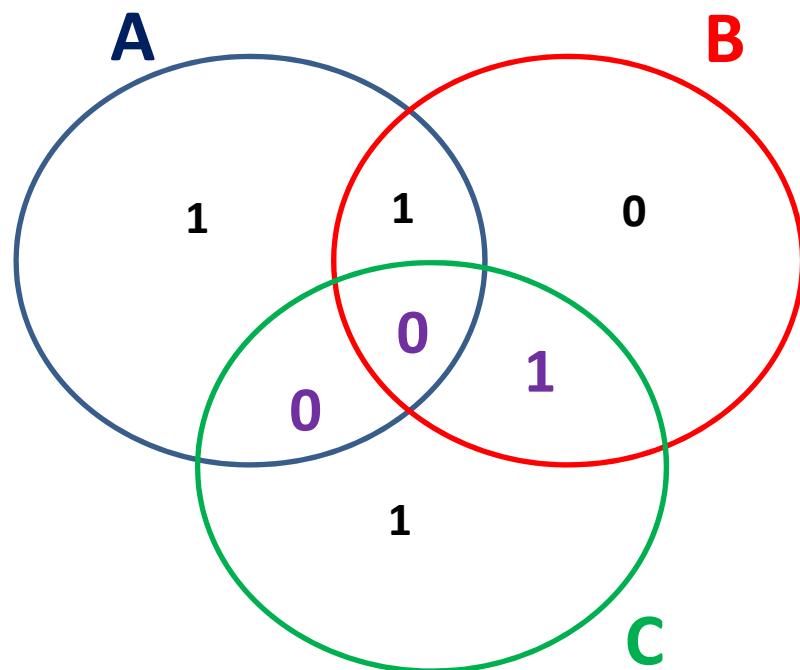
Seja uma palavra válida = 1101 colocada no diagrama de Venn a seguir:



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

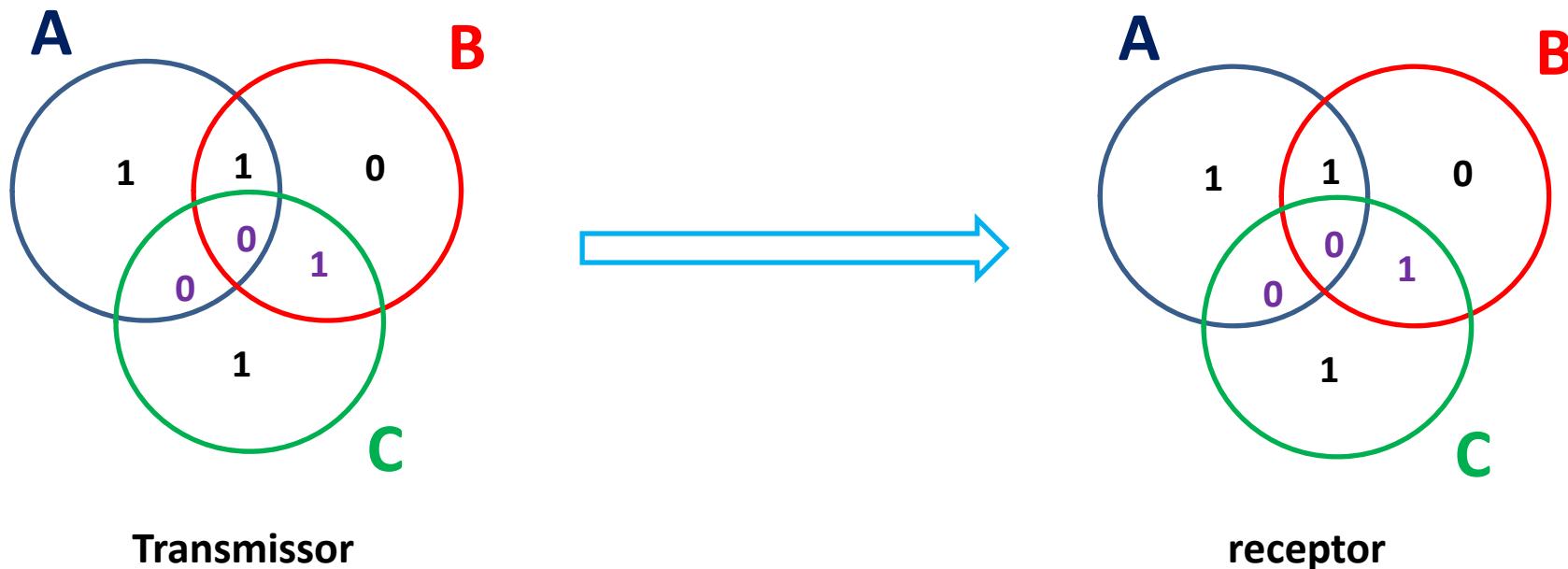
Seja uma palavra válida = 1101 colocada no diagrama de Venn a seguir:



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:

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

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 $M=6 \Rightarrow r=4$

\vdots
 $M=11 \Rightarrow r=4$ 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 $0 =$ na fórmula!

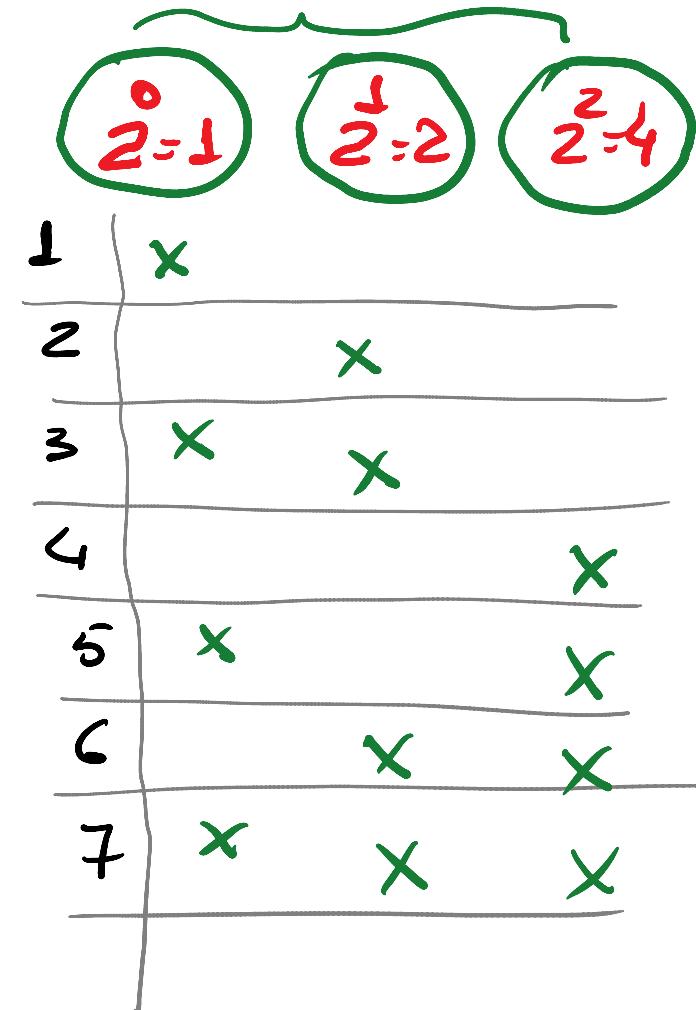
2) Onde colocar os bits?

Exemplo: palavra valida = 1101 (ou $13_{(10)}$)

Como $m=4 \Rightarrow r=3$

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

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



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?

<u>1 0 1 0 0 0 1</u>						
1	2	3	4	5	6	7
1	1	0	/			
1.	3.	5.	7	erro		
0	1	0	/			
2.	3.	6.	7	OK		
0	0	0	/			
4.	5.	6.	7	erro		

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

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

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?