

Arquitetura de Computadores

Aula 04

Códigos de correção de erro: Códigos Hamming

2.2.4 Códigos de correção de erro

Motivação

- **Erros ocorrem:** 0's se transformam em 1's devido a causas espúrias em dispositivos de memória, em sistemas de comunicação, etc.
- **Objetivo:** Procurar mecanismos que sejam capazes de detectar ou, melhor ainda, corrigir este erros.

2.2.4 Códigos de correção de erro

- ❑ Memórias de computador podem cometer erros de vez em quando devido a picos de tensão na linha elétrica ou outras causas.
- ❑ Para se resguardar contra esses erros, algumas memórias usam **códigos de detecção** de erros ou **códigos de correção** de erros.
 - Para isso bits extras são adicionados a cada palavra de memória de modo especial.

2.2.4 Códigos de correção de erro

Palavra: m bits de dados + r bits redundantes (ou de verificação)

Comprimento da palavra = n

Logo $n = m + r$

Palavra de código de n bits: uma unidade de n bits que contém m dados e r bits de verificação.

2.2.4 Códigos de correção de erro

Sua principal significância é que, se duas palavras de código estiverem separadas por uma distância de Hamming d , será preciso d erros de bits únicos para converter uma na outra.

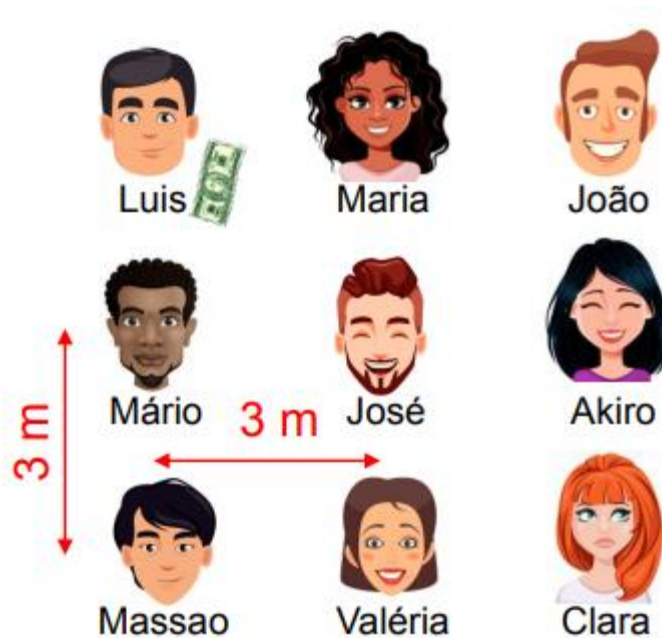
10001001 10110001 -> 3 bits diferentes

1	\oplus	1	0
0	\oplus	0	0
0	\oplus	1	1
0	\oplus	1	1
1	\oplus	0	1
0	\oplus	0	0
0	\oplus	0	0
1	\oplus	1	0



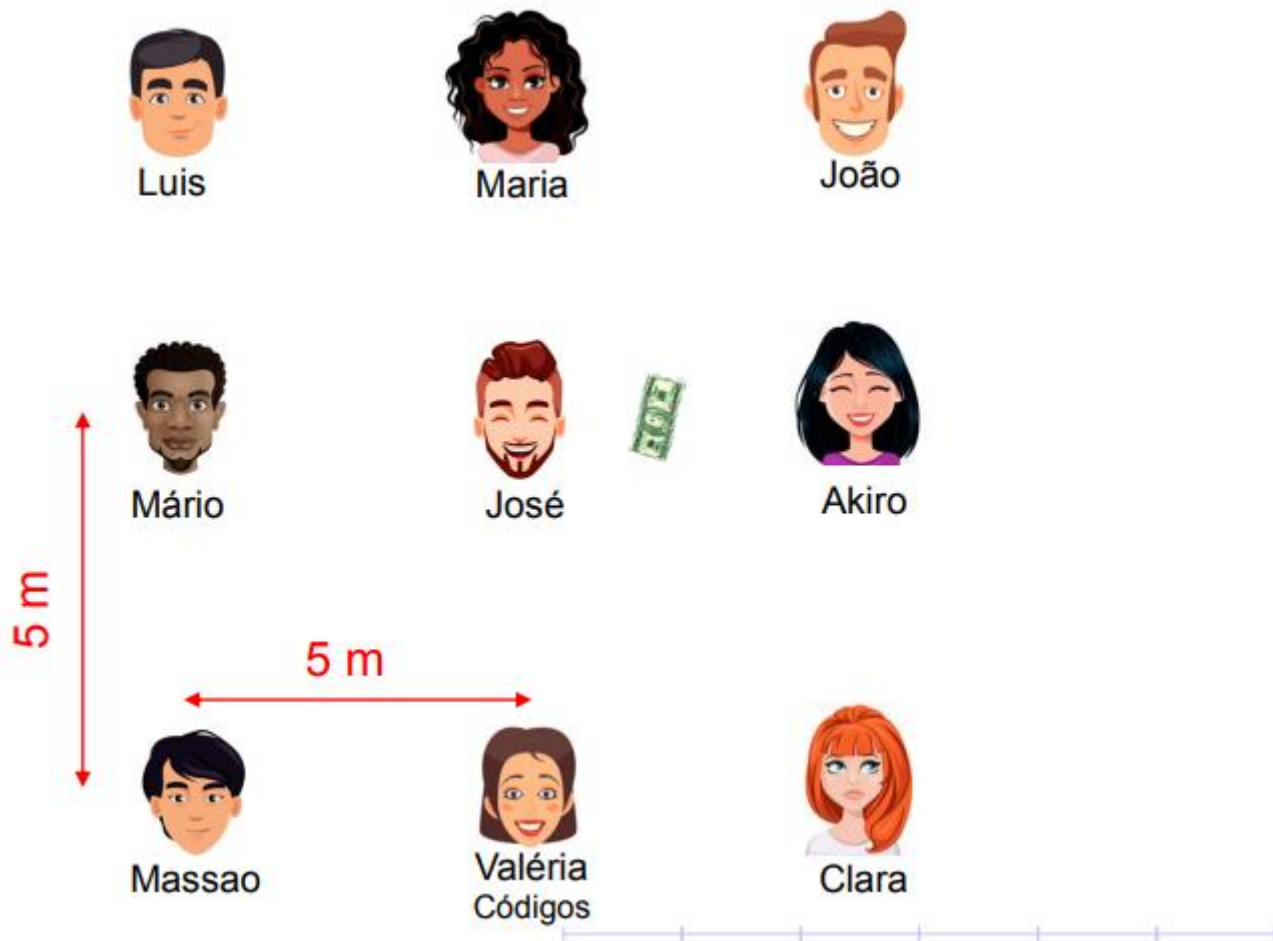
**Distância de
Hamming**

2.2.4 Códigos de correção de erro



De quem é?

2.2.4 Códigos de correção de erro



2.2.4 Códigos de correção de erro

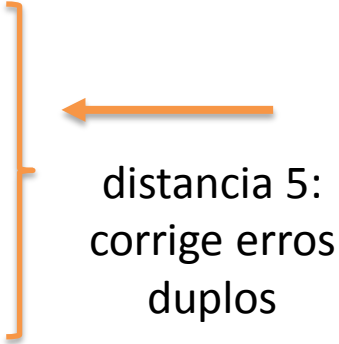
As propriedades de detecção de erro e correção de erro de um código dependem de sua distância de Hamming.

- Para **detectar** d erros de bits únicos, é preciso de um código de distância $d+1$
 - Com tal código, não existe nenhum modo que permita que d erros de bits únicos mudem uma palavra de código válida para uma outra palavra válida
- Para **corrigir** d erros de bits únicos, é preciso de um código de distância $2d+1$
 - Desse modo, as palavras de código legais estão tão distantes uma da outra que, mesmo que d mude, a palavra de código original ainda estará mais perto do que qualquer outra palavra de código, portanto ela pode ser unicamente detectada

2.2.4 Códigos de correção de erro

Exemplo

Considere um código que tenha apenas quatro palavras de código válidas:

- 0000000000
 - 0000011111
 - 1111100000
 - 1111111111
- 
- distancia 5:
corrige erros
duplos

Chegou ao destinatário a seguinte palavra:

0000000111

Qual deve ter sido a palavra correta enviada?

R-> 0000011111

↑
Isso para erro duplo

Mas para erro triplo não poderemos corrigir

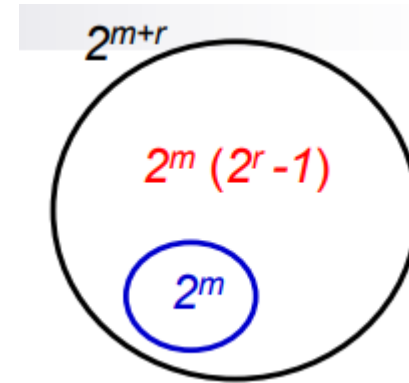
2.2.4 Códigos de correção de erro

Ideia Básica

- Para representar 2^m palavras utilizam-se $n=r+m$ bits (r bits além do necessário!)

Entre as

- 2^{m+r} palavras possíveis
- 2^m palavras são válidas e
- $2^m (2^r - 1)$ são inválidas.



- O sistema só gera palavras válidas.
- Logo, se surgir alguma palavra inválida, trata-se de um erro!

2.2.4 Códigos de correção de erro

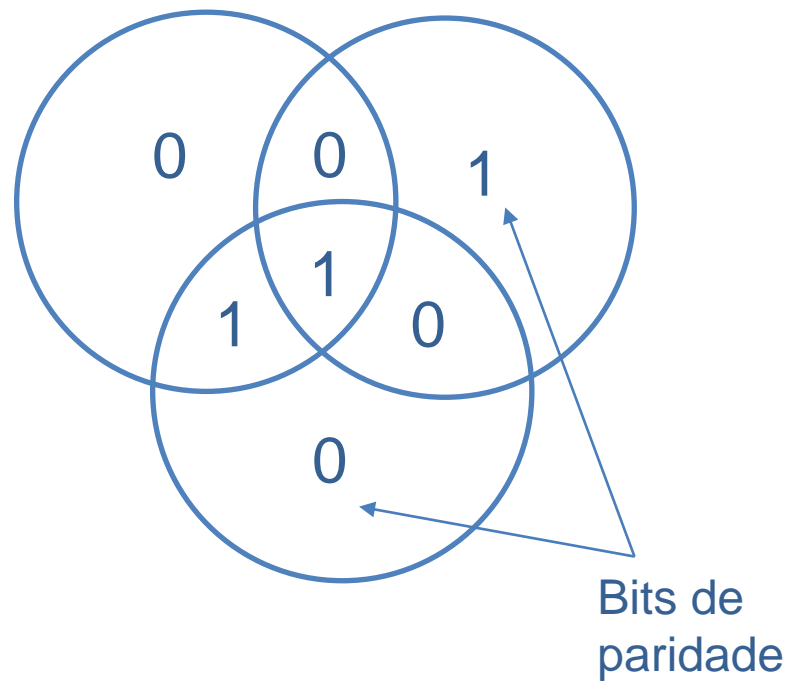
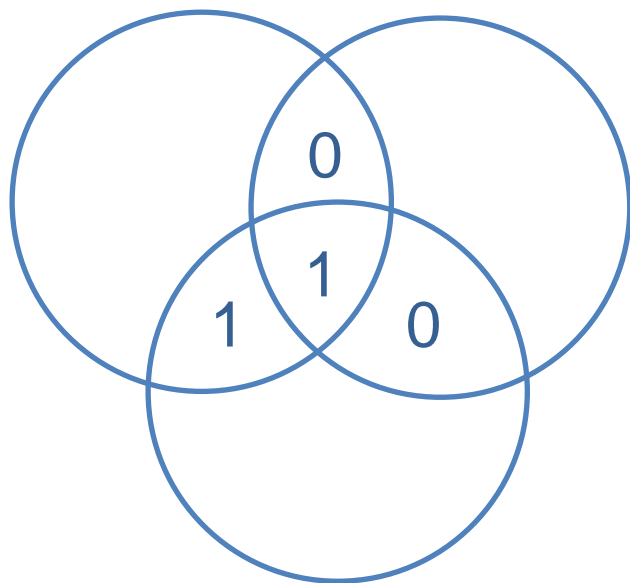
- Para **detectar** d erros de bits únicos, é preciso de um código de distância $d+1$
- Para **corrigir** d erros de bits únicos, é preciso de um código de distância $2d+1$

Tam. da palavra	Bits de verificação	Tam. 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	256	4
512	10	522	2

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

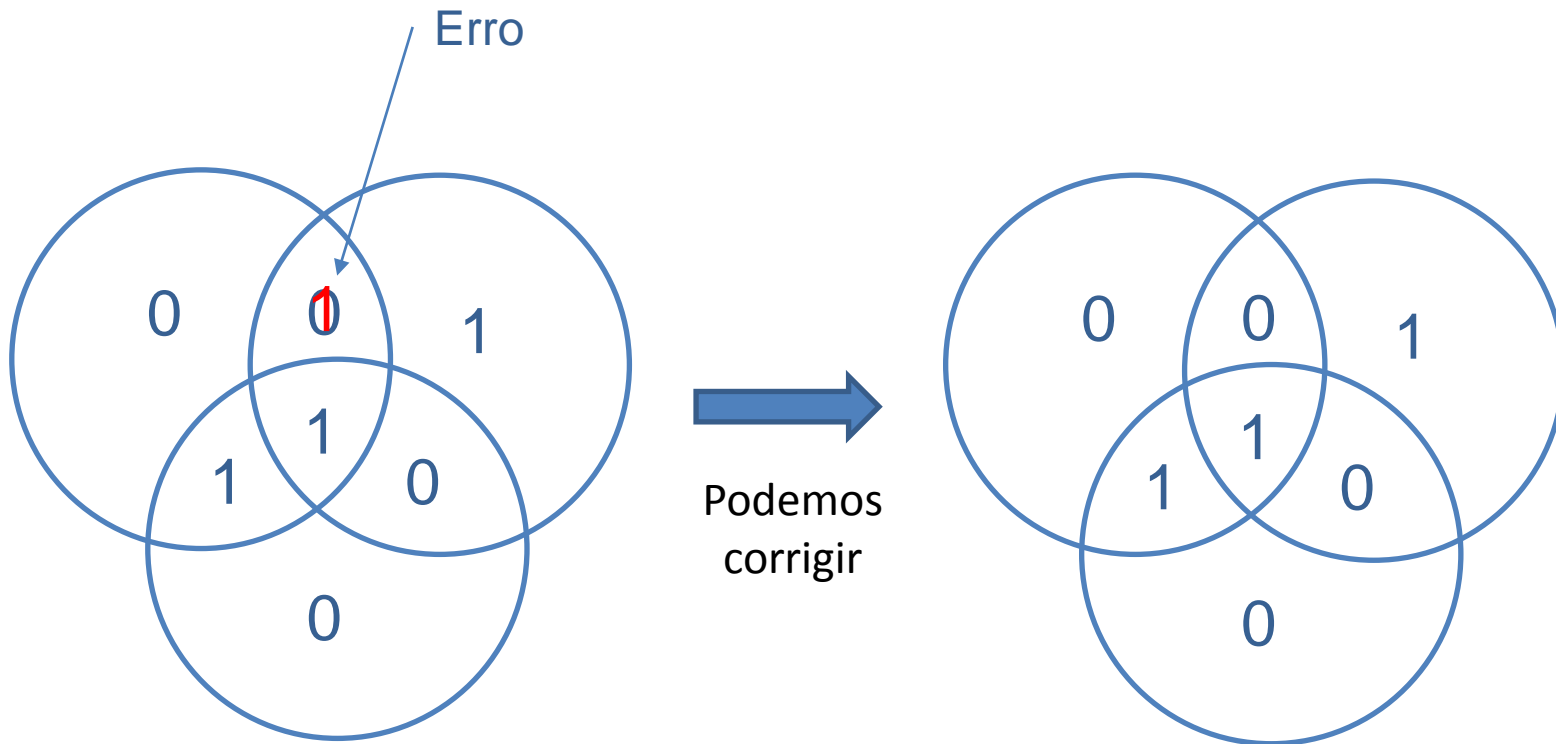
2.2.4 Códigos de correção de erro

Paridade



2.2.4 Códigos de correção de erro

Paridade



2.2.4 Códigos de correção de erro

Como calcular o código de Hamming...

Algoritmo:

1

Crie a palavra de dados. Qualquer bit com uma posição que for uma potência de dois (primeiro, segundo, quarto, etc.) deve ser reservado para informações de paridade. Use o tamanho que for necessário para que a palavra tenha os dados originais e os bits de paridade.

Exemplo: 1 1 0 1 0 0 1 0 transforma-se em _ _ 1 _ 1 0 1 _ 0 0 1 0

Os bits originais permanecem na mesma ordem, mas foram espalhados para inserirmos os bits de paridade.

2.2.4 Códigos de correção de erro

2

Calcule o primeiro bit de paridade. Começando com o primeiro bit, lê-se um bit e, em seguida, pula-se um bit e repete-se o procedimento até o final. Enquanto isso conta-se o número de uns encontrado. Os bits de paridade não contam neste processo. Se o número de uns for par, defina o primeiro bit como zero. Caso contrário, defina-o para um.

Exemplo: Bits 1, 3, 5, 7, 9 e 11 de _ _ 1 _ 1 0 1 _ 0 0 1 0, _11101, contêm quatro uns. Este é par, então, o primeiro bit é definido como zero: 0 _ 1 _ 1 0 1 _ 0 0 1 0

2.2.4 Códigos de correção de erro

3

Calcule os bits de paridade restantes. Começando com o segundo bit, lê-se dois bits e, em seguida, pula-se dois bits e repete-se o procedimento até o final. O quarto bit lê quatro bits, pula outros quatro, começando pelo bit quatro. O mesmo padrão é seguido por todos os bits de paridade, até todos serem computados.

Exemplo: Bit 2: 0 _ 1 _ 1 0 1 _ 0 0 1 0 verifica _1, 01, 01, que contêm três uns, então o bit 2 é definido como um. Bit 4: _ 0 1 1 1 0 1 _ 0 0 1 0 verifica _101, 0, que contêm dois uns, então o bit 4 é definido como zero. Bit 8: 0 1 1 0 1 0 1 _ 0 0 1 0 verifica _0010, que contêm só um, então o bit 8 é definido como um. A palavra é, portanto, codificada como 011010110010.

4

Confirme a palavra. Se uma palavra estiver corrompido, os bits de paridade não vão coincidir com o que é esperado. Para confirmar que a palavra não esteja corrompida, basta calcular os bits de paridade usando as etapas dois e três. Se os bits não forem iguais, grave suas posições.

5

Corrija o bit errado. Se você encontrar bits de paridade incorretos, simplesmente some as posições dos bits. O valor da soma é a posição do bit incorreto. Troque o valor do bit nesta posição. Por exemplo, se os bits de paridade incorretos forem o um e o quatro, trocar o valor do quinto bit corrigirá o erro.

2.2.4 Códigos de correção de erro

Como calcular o código de Hamming...

Exemplo:

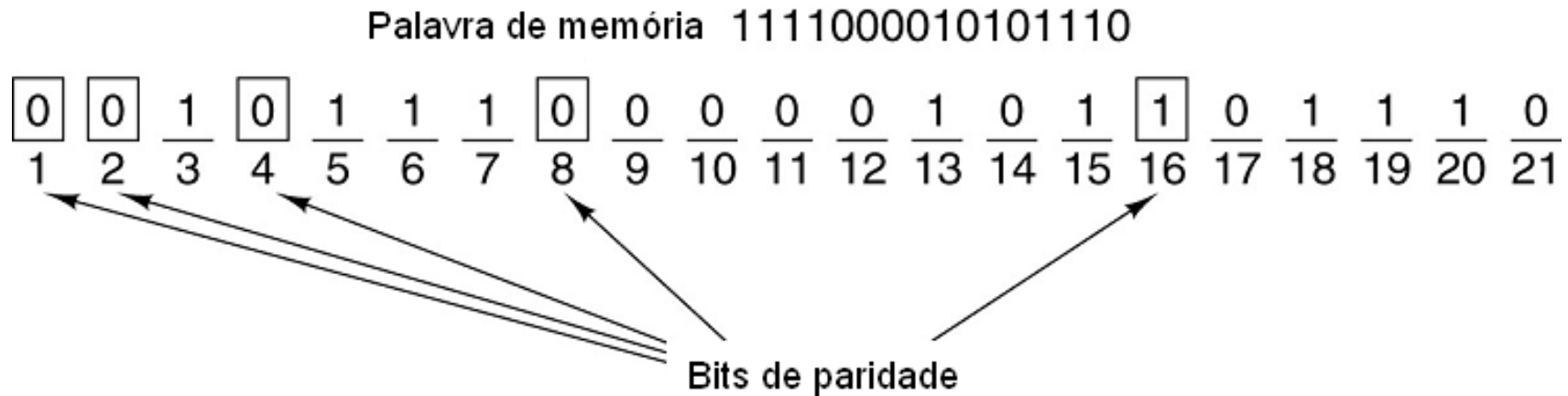
certo 1000001

dados: 00101001001



Copie do quadro

2.2.4 Códigos de correção de erro



Construção do código de Hamming para a palavra de memória 1111000010101110 adicionando 5 bits de verificação aos 16 bits de dados.

- Bit 1: verifica bits 1, 3, 5, 7, 8, 11, 13, 15, 17, 19, 21
- Bit 2: verifica bits 2, 3, 6, 7, 10, 11, 14, 15, 18, 19
- Bit 4: verifica bits 4, 5, 6, 7, 12, 13, 14, 15, 18, 19
- Bit 8: verifica bits 8, 9, 10, 11, 12, 13, 14, 15
- Bit 16: verifica bits 16, 17, 18, 19, 20, 21

Exercícios

1. Deseja-se construir um código composto de 3 palavras código de 8 bits cada uma. As palavras código devem ser escolhidas de modo a maximizar o número de erros simples que se poderá detectar. Pergunta-se quantos erros simples se poderá corrigir com tal código?
2. Derive uma relação que forneça um limite inferior para o número de bits de verificação (r) necessários para se montar um código de correção de até 2 erros simples para 2^m palavras válidas distintas.

2.2.4 Códigos de correção de erro

Exercícios

3. Seja o código formado pelas palavras código abaixo:

10100110 - 11111111 - 01011001 - 10000101

Qual é a distância de Hamming do Código? Qual é o número máximo de erros simples que se pode detectar neste código? Qual é o número máximo de erros simples que se pode corrigir neste código?

2.2.4 Códigos de correção de erro

Exercícios

4. Qual é o código completo contendo 12 bits produzido pelo código de Hamming derivado palavra de 8 bits abaixo?

1 0 1 0 0 1 0 1

5. Altere aleatoriamente um dos bits do resultado à questão anterior e verifique se você consegue identificar o bit alterado, e portanto, corrija-lo.