Bacharelado em Ciência da Computação **UNIOESTE**
Colegiado de Ciência da Computação - CCET Campus Cascavel - Rua Universitária, 1619

Disciplina: Organização e Arquitetura de Computadores **Data:** 10/05/2018
Série: 3 **Prática:** 02 - Hamming

Conteúdo

INTRODUÇÃO

ERROS em palavras:

Falha Permanente: Defeito físico que afeta células de memória ou (sinais de transmissão) de modo a alterar seu funcionamento, tornando-as não-confiáveis para armazenamento.

Erro não Permanente: É um evento aleatório e não destrutivo que altera o valor lógico de uma célula de memória (ou sinal de transmissão) sem causar dano físico.

As falhas e Erros são indesejáveis, mas acontecem frequentemente. Assim, códigos para detecção de erros (EDC) e códigos de detecção e correção de erros (ECC) são implementados nos mais diversos sistemas.

EDC	**Descrição**
Paridade	Apenas 1 bit extra
CRC	Redundância Cíclica (Polinômio $G(x)$)
Checksum	Algoritmos Hash de Segurança
ECC	**Descrição**
Hamming	Paridade de conjuntos de bits da palavra

BIT DE PARIDADE

Utiliza apenas um bit **R** de redundância que é composto pela função lógica OU_Exclusivo aplicado em todos os bits da palavra **P.** O bit **R** é gravado (ou transmitida) juntamente com a palavra **P.**

Exemplo: P = $_{msb}$**01101101**

Assim: R = **0** xor **1** xor **1** xor **0** xor **1** xor **1** xor **0** xor **1** = **1**

Palavra Gravada/Transmitida: $_{msb}$**011011011**

CRC

Cyclic Redundancy Check (CRC) é utilizado em redes de comunicação e armazenamento para **detectar** eventuais alterações, propositais ou oriundas de falhas de sistema, nos bytes dos dados.

Em Linux, existe a ferramenta cksum que permite gerar o código CRC para comparar com os códigos fornecidos pelas fontes dos dados. Se uso é simples: basta usar o comando:

```
$:> cksum <nomeDoArquivo>
```

Checksum

Mais conhecidos como Algoritmos Hash de Segurança, tem o mesmo objetivo do CRC, porém é mais aceito na internet. Existem diversos algoritmos: como o MD5 (Message-Digest algorithm 5) e o SHA-2 (Secure Hash Algorithm 2), sendo este último, um algoritmo que gera um HASH de 256 bits.

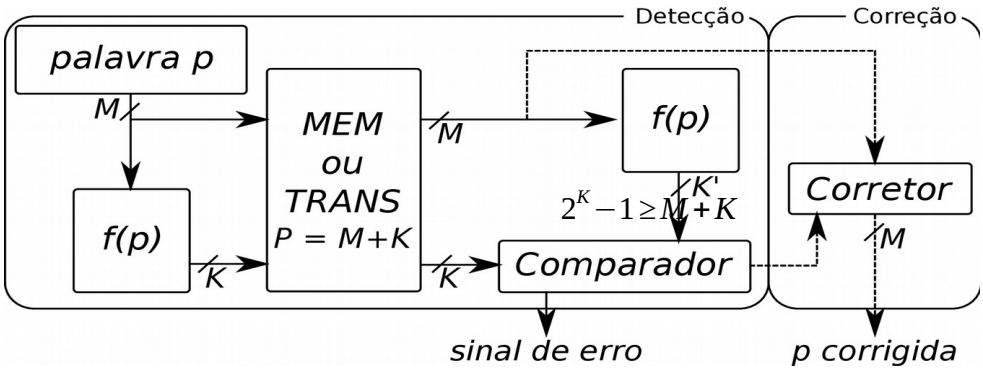
Normalmente, servidores de arquivos disponibilizam o resultado do SHA-2 em arquivos com formatação do tipo:

<HASH> *<nomeDoArquivo>

Para verificar, usa-se o comando:

$:> sha256sum -c <nomeDoArquivoFormatadoComHASH+NomeArquivo>

FUNCIONAMENTO DO PROCESSO DE DETECÇÃO E CORREÇÃO



CÓDIGO DE HAMMING – SEC (Págs. 136~140 de STALLINGS 8ª Ed.)

Single Error Correction

Em uma palavra de **8** bits precisa-se de **4** bits para detecção de erros, totalizando 12 bits para a palavra final:

Determinação do número de bits é dado por:

Onde:

K = Tamanho da palavra de correção

M = Tamanho da palavra

Desta forma:

$$2^4 - 1 \geq 8 + 4 \rightarrow 15 \geq 12$$

Disposição dos bits para palavra de 8 bits + ECC:

- Dados: **M**
- Teste: **C**

Tabela 1: Disposição dos bits de Teste **C** e Dados **M**.

12	11	10	9	8	7	6	5	4	3	2	1
M8	M7	M6	M5	C8	M4	M3	M2	C4	M1	C2	C1
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001

Bits de Dados **M** válidos para cálculo dos bits de teste **C**:

Tabela 2: Bits de Teste **C** e seus respectivos bits de Dados **M**.

C1	**M1**	**M2**		**M4**	**M5**		**M7**	
C2	**M1**		**M3**	**M4**		**M6**	**M7**	
C4		**M2**	**M3**	**M4**				**M8**
C8					**M5**	**M6**	**M7**	**M8**

Nota-se que, cada bit teste **C** calcula a paridade dos demais bits cuja posição contenha o valor 1 correspondente a este:

Tabela 3: Exemplo de equivalência entre o bit de teste **C1** e os bits de Dados **M**.

C1	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
0001	**0011**	**0101**	**0110**	**0111**	**1001**	**1010**	**1011**	**1100**

Passos:

Durante a Escrita:

1 – Cálculo do bit de teste **C**: **Paridade** dos bits de dados **M** respeitando a tabela 2;

2 – Aloca os bits de dados **M** e de teste **C** na mesma palavra.

Durante a Leitura:

1 – Lê-se a palavra completa, e extrair os bits de Dados **M** e Teste **C**;

2 – Cálculo do novo bit de teste **C'**;

3 – Compara-se o bit de teste **C** com novo bit de teste **C'** (com **ou-exclusivo**);

4a – Palavra síndrome diferente de 0 e menor do que M:

Dados **M** incorreto -> Identificar e recuperar;

Teste **C** incorreto -> Assume-se que os bits **M** estejam corretos.

4b – Palavra síndrome maior do que M

Dados **M** incorreto e impossível de recuperar (Rejeição Automática)

4b – Palavra síndrome igual a 0: Dados **M** correto.

CÓDIGO DE HAMMING – SEC-DED

Single-Error-Correction-Double-Error-Detection

Adiciona-se um bit extra **G** que representa a paridade geral da palavra, evitando piorar a situação caso a palavra contenha mais do que um único erro.

Tabela 4: Posicionamento do bit **G**, dos bits de teste **C** e dos bits de Dados **M**.

12	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
M8	**M7**	**M6**	**M5**	**C8**	**M4**	**M3**	**M2**	**C4**	**M1**	**C2**	**C1**	**G**
1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	1101

Gravação:

- O bit **G** é calculado após o cálculo da palavra **C**, em conjunto com **M**.

Aplicação de ou-exclusivo (bit de paridade!)

Leitura:

- Após o processo de recuperação SEC, calcula-se um novo bit **G'**

Compara-se **G** com **G'**

Se forem iguais, a palavra **M** está correta

Se forem diferentes, 2 erros aconteceram e **SEC** piorou a situação

Palavra **M** deve ser Rejeitada!

Simulador Hamming SEC-DED

Grupos: Individual ou dupla

Linguagem: Livre

Forma de Avaliação: Defesa por grupo

Data: à combinar (em horário de Aula Prática)

O grupo deve elaborar um simulador de armazenamento seguro (utilizando SEC-DEC) de arquivos binários:

Funcionamento para gravação:

a) O arquivo para armazenamento deve ser passado via parâmetro no prompt/terminal

a1) ou ser possível de selecionar nos simuladores com interface gráfica

b) Deve ser mostrado os principais passos para produção da palavra síndrome + bit G (M = 8)

c) Deve ser salvo um novo arquivo com SEC-DED no formato da tabela 4

Exemplo:

```
$ ./hamming arquivoTeste.qqcoisa -w
```

-w para salvar o arquivo com integridade hamming

Sugestão de extensão para o arquivo com integridade: .sotw

Funcionamento para leitura:

d) O arquivo para leitura deve ser passado via parâmetro no prompt/terminal

d1) ou ser possível de selecionar nos simuladores com interface gráfica

e) Deve ser mostrado os principais passos para verificação de integridade do arquivo

f) Em caso de falha, executar os procedimentos necessários para correção

f.1) Salvar o arquivo corrigido

g) Finalizar

Exemplo:

```
$ ./hamming arquivoTeste.wham -r
```

-r para ler o arquivo com integridade hamming, verificar e recuperar (se necessário)