Relatório Laboratório 1 – Parte 2: Implementação e Análise do CRC

Parte 3 – Análise Comparativa de Desempenho:

1. Objetivo

O seguinte relatório tem como objetivo realizar uma análise comparativa de desempenho entre a implementação manual do algoritmo crc feita pelo grupo e a implementação pronta presente na biblioteca indicada pelo professor.

As métricas comparativas analisadas foram o tempo de execução e o pico do uso de memória de ambas as implementações, executadas na máquina explicitada, os testes foram feitos para os tamanhos de 1500, 3000, 6000 e 16000 bytes.

2. Máquina Utilizada (Ambiente de Teste)

Processador: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

Núcleos / Threads: 4 / 8

RAM: 4 GB

SO: Microsoft Windows 11 Home Single Language

3. Metodologia

As implementações manual e da biblioteca foram executadas para os tamanhos de 1500, 3000, 6000 e 16000 bytes, utilizando bits aleatórios, para cada execução o tempo foi medido utilizando time.perf_counter(), como também foram medidos os picos de memória utilizando tracemalloc.

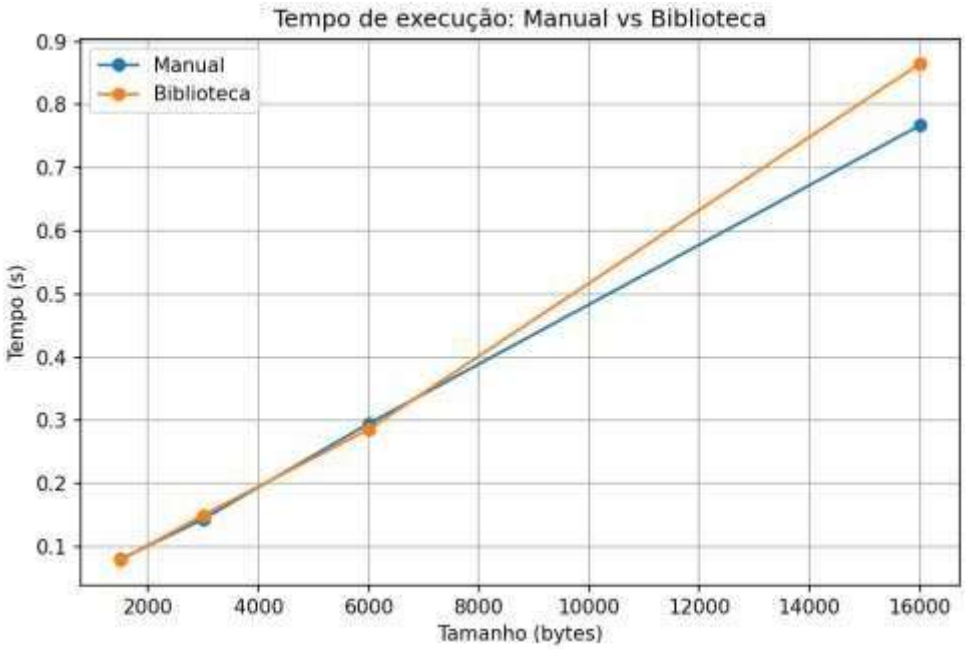
4. Resultados

4.1 Tabela Comparativa

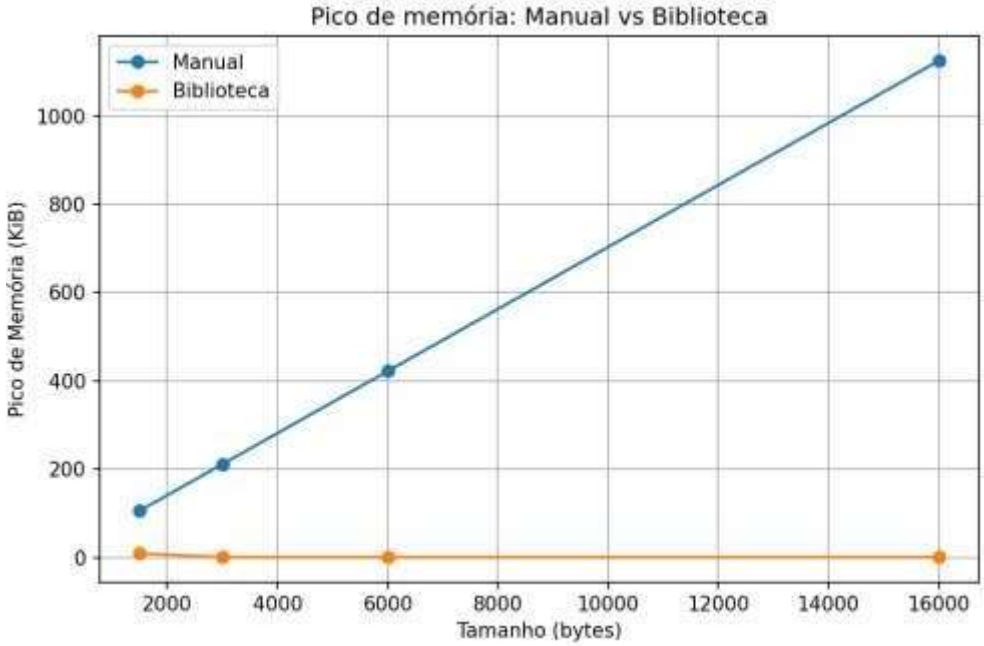
Tamanho (bytes)	Tempo Manual	Tempo Biblioteca	Pico Mem. Manual (KiB)	Pico Mem. Lib (KiB)
1500	0.080	0.078	105.758	8.756
3000	0.143	0.149	211.250	0.675
6000	0.294	0.285	422.188	0.675
16000	0.766	0.864	1125.313	0.675

4.2 Gráficos

4.2.1 Tempo x Tamanho



4.2.2 Pico Memoria x Tamanho



5. Análise e Discussão

5.1 Tempo

Os dados observados indicam que tanto a implementação manual realizada pelo grupo quanto a implementação da biblioteca possuem um tempo de execução da mesma ordem de grandeza, mostrando boa eficiência temporal da implementação manual, uma possível motivação é que tanto a biblioteca quanto a implementação do grupo utilizam python puro, tendo assim tempos no geral similares.

5.2 Memória

Os dados obtidos evidenciam que a implementação manual tem um pico de memória expressivamente maior que a implementação da biblioteca, tempo um aumento conforme o tamanho da mensagem, as implementações possuem ordens de grandeza diferentes, possivelmente está ineficiência da implementação manual vem da alocação temporária de muitas estruturas como listas e strings. Por outro lado, a biblioteca mantém um pico de memória baixo praticamente constante.

6. Conclusão

Observados os dados obtidos podemos concluir que a implementação manual cumpriu com o quesito de corretude, sendo também uma ótima escolha quando um dos requisitos é baixo tempo de execução para mensagens grandes, porém pecou na utilização de memória, tendo picos que crescem linearmente com o tamanho da mensagem, sendo relativamente ineficiente quando comparado com a implementação da biblioteca indicada pelo professor que possui um tempo de execução semelhante e por vezes menor, como também picos de memória praticamente constantes.

Parte 4 – Análise Investigativa da Detecção de Erros:

1. Configuração do Cenário

MENSAGEM: Lucas Andrade Souza

MENSAGEM_BASE (bits ASCII):
0100110001110101011000110110000101110011001000000 1...0100100000010100110110111101110101011110100110000 1

Tamanho da mensagem: 152 bits

Matrícula final: 7

Gerador usado: CRC-16/MAXIM

Gerador (bits): 10011000000010001

CRC calculado: 0111101110001101

2. Exemplos de Erros não Detectados

Exemplo 1:

Posição: 55

Tamanho do burst: 17 bits

Bits originais: 10110111001100100

Bits corrompidos: 01001000110011011

Padrão de erro: 11111111111111111

Exemplo 2:

Posição: 19

Tamanho do burst: 17 bits

Bits originais: 00011011000010111

Bits corrompidos: 11100100111101000

Padrão de erro: 11111111111111111

Exemplo 3:

Posição: 117

Tamanho do burst: 17 bits

Bits originais: 01101101111011101

Bits corrompidos: 10010010000100010

Padrão de erro: 11111111111111111

Como esperado, erros com burst > 16 bits (grau do polinômio) podem escapar da detecção. A probabilidade de não detecção para bursts > r é aproximadamente 2^(-r), ou seja, cerca de 1 em 65.536 para CRC-16. Os erros não detectados ocorrem quando o padrão de erro é divisível pelo polinômio gerador, resultando em resto zero. Isso demonstra a importância de escolher o polinômio gerador adequado e entender as limitações do CRC para o tipo de erro esperado.

3. Resultados dos 10 Testes

Teste	Posição	Tamanho Burst	Detecção	Padrão de Erro
1	28	7	SIM	0001011 -> 1110100
2	70	2	SIM	00 -> 11
3	57	3	SIM	110 -> 001
4	26	3	SIM	100 -> 011
5	139	7	SIM	1101001 -> 0010110
6	151	9	SIM	101111011 -> 010000100
7	8	14	SIM	01110101011000 -> 10001010100111
8	23	8	SIM	10110000 -> 01001111
9	59	11	SIM	01110011001 -> 10001100110
10	6	16	SIM	0001110101011000 -> 1110001010100111

4. Conclusões

O CRC-16/MAXIM tem capacidade de detectar:

- Todos os erros de bit único
- Todos os erros de burst até 16 bits
- Todos os erros com número ímpar de bits

2. Todos os erros foram detectados nestes testes, mas isso não significa que o CRC é perfeito. Com mais testes ou bursts maiores, eventualmente encontraríamos padrões não detectados.

3. Limitações conhecidas do CRC:

- Não pode detectar 100% dos erros para bursts > r bits
- A probabilidade de não detecção é aproximadamente 2^(-r) para erros aleatórios
- Para CRC-16, isso significa ~0.0015% de chance de falha