JoSDC'24 Final Benchmarks

Benchmark 3

Decryption

Author: Issa Qandah

1. Problem Description

This benchmark evaluates a simplified Feistel network-based decryption algorithm operating in Cipher Block Chaining (CBC) mode. Participants are required to decrypt a given pre-encrypted ciphertext and correctly recover the original plaintext.

1.1. Feistel Network: Concept and Functionality

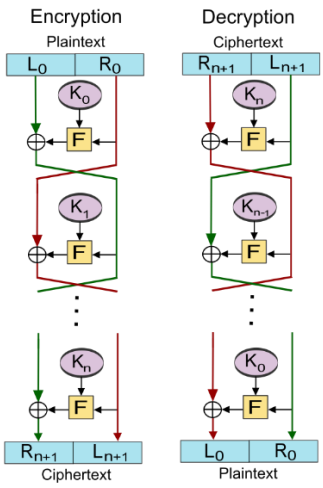
A Feistel network is a symmetric structure widely used in block ciphers (e.g., DES), allowing encryption and decryption with the same core function. It involves multiple rounds where data is split into two halves and transformed through a round function.

Feistel Network Process:

1. **Split Data into Two Halves:**
   - A 64-bit input block is divided into left (L) and right (R) halves, each 32 bits.
2. **Apply Round Function:**
   - The right half (R) is processed through a round function (F).
   - The output of F(R, Key) is XORed with the left half (L).
3. **Swap Halves:**
   - Left half is replaced by the previous right half (R → L).
   - The new right half is set to the XOR result (L ⊕ F(R, Key) → R).
4. **Repeat for Multiple Rounds:**
   - This process continues for 150 rounds to ensure strong encryption.
5. **Final Round (Decryption Considerations):**
   - Decryption reverses the encryption steps using the same round function.

Feistel Function in This Benchmark:

- XOR the right half with the encryption key.
- Perform a right bitwise shift by 3 bits.

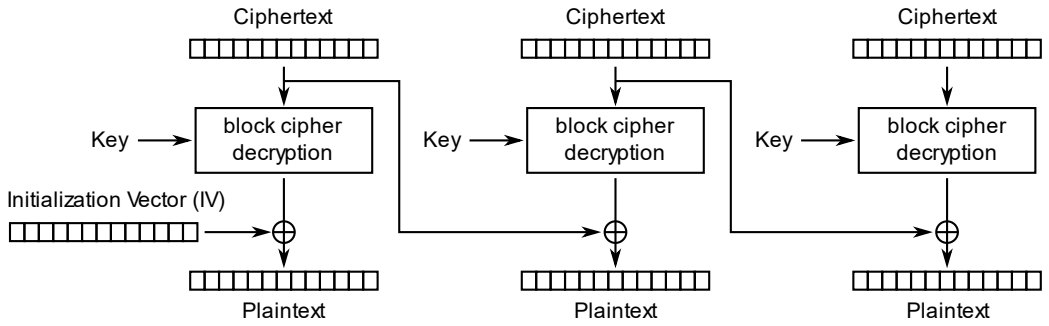


1.2. Cipher Block Chaining (CBC) Mode

CBC mode enhances security by introducing an Initialization Vector (IV) and linking each block's encryption to the previous block.

CBC Decryption Process:

1. **Initialize IV:**
   - The first block is XORed with a random IV before decryption.
2. **Decrypt Each Block Using Feistel Network:**
   - Each encrypted block undergoes Feistel decryption.
3. **Apply CBC Chaining:**
   - After decryption, the output is XORed with the previous ciphertext block (or IV for the first block).
   - This step restores the original plaintext.
4. **Update IV:**
   - The current ciphertext block is stored and used as the IV for the next block.



Cipher Block Chaining (CBC) mode decryption

Step-by-Step Benchmark Execution:

1. Read the Initialization Vector (IV) from memory.
2. Loop through each encrypted block:
   - Load the ciphertext block.
   - Store it temporarily for IV update.
   - Perform Feistel decryption with 150 rounds.
   - XOR the result with the IV (CBC mode).
   - Store the decrypted block in memory.
   - Update IV with the original ciphertext block.
3. Repeat until all blocks are processed.
4. Exit the program.

2. Decryption Algorithm

Parameters:

- **B (Number of Blocks):** 1000 blocks (each block = 2 words)
- **R (Number of Rounds):** 150 rounds

Pseudocode:

```
# CBC Mode Feistel Decryption - Pseudocode

Initialize:
    - Set numBlocks = 1000, numRounds = 150.
    - Load IV and encryption key.
    - Set pointers for ciphertext and decrypted text.

Decrypt Loop (for each block):
    1. Load ciphertext block (L, R).
    2. Save current ciphertext as IV for next block.
    3. Perform Feistel decryption:
       - Repeat for numRounds:
         - Apply Feistel function: XOR L with key, shift right.
         - Compute new L: R XOR Feistel result.
         - Swap L and R.
    4. XOR (L, R) with IV for CBC decryption.
    5. Store decrypted block.
    6. Update IV and move to the next block.

End:
    - Exit after all blocks are decrypted.
```

3. Specifications and Complexity Analysis

3.1 Storage Complexity

This benchmark requires the following memory allocations:

Variable	Size (Words)
Key	1
IV	2
Ciphertext	2000
Decrypted Text	2000
Total Storage	**4003 Words**

For practical allocation, a **4KW (4096 words) memory space** is recommended, ensuring sufficient room for additional data.

Memory Layout:

Variable	Start Address	End Address
Key	0	0
IV	1	2
Ciphertext	3	2002
Decrypted Text	2003	4002

Test Data Used:

- The ciphertext is pre-encrypted using the same algorithm, key, IV, and rounds.
- After decryption, the expected output should be sequential numbers from **1 to 2000**.

3.2 Instruction Memory

- **Total MIPS Instructions:** 31
- **Storage per Instruction:** 1 word
- **Instruction Memory Requirement:** 256 words (ample for execution)

Notes for Assembly Processing:

- If modifications (e.g., NOPs, reordering, or optimizations) are made by the assembler, they must be documented.

3.3 Time Complexity

The decryption time complexity is:

O(B × R)

Where: **B = 1000 blocks**

R = 150 rounds per block

4. Related Attached Files

The benchmark directory includes the following files:

File Name	Functionality
Decryption.asm	MIPS assembly code, structured with guiding comments to ensure clarity and ease of understanding
definition.txt	Defines the .data section for memory allocation.
memoryInit.txt	Contains hexadecimal values for the key, IV, and ciphertext.
dataMemoryWORD.mif	Memory initialization file for word addressing mode.
dataMemoryWORD.hex	Hex format memory initialization file (word addressing mode).

Important Notes

- If using a different addressing mode, adjust your memory initialization file accordingly.
- Submit machine code files along with execution results and any required documentation for benchmark processing and evaluation.