1. Understanding the Task

The project requires me to encrypt and decrypt a secret one-word English message using the XOR logical operator, as covered in the lectures. The key must be a one-word secret of the same length as the message. I must perform character-by-character XOR operations in a loop and save the plaintext, key, encrypted text, and decrypted text in a text file named output.txt.

2. Data Setup

I begin by declaring my secret plaintext and key in the .data section, ensuring they are equal in length. In this project, I selected "HELLO" as the plaintext and "world" as the key. I allocate buffers of the same size to hold the encrypted and decrypted results. Along with the data, I prepare the labels for each output section, such as "Plain text:", "Key:", "Encrypted text:", and "Decrypted text:". These labels will help format the output file clearly. I also specify the filename output.txt.

3. Encryption Implementation

Using the lectures on logical instructions and loops, I implement the encryption with a loop that runs once for each character of the plaintext/key.

- I load one character from the plaintext and one from the key.
- I perform a bitwise XOR operation on these two characters.
- The result is stored in the encrypted buffer.
- I increment pointers and repeat until all characters are processed.

This loop uses only the registers and instructions demonstrated in class, respecting the project constraints.

4. Decryption Implementation

Because XOR is reversible, the decryption process uses the same loop logic. This time, the inputs are the encrypted buffer and the original key.

- Each encrypted character is XORed with the corresponding key character.
- The result is stored in the decrypted buffer.
- After processing all characters, the decrypted text matches the original plaintext, confirming correctness.

5. Writing to Output File

From the lectures on memory organization and system calls, I know how to open, write, and close files using Linux syscalls (int 0x80). I open output.txt with create, write-only, and truncate flags, ensuring the file is ready to be overwritten on each run.
I then write each section to the file in order:

1. Write the "Plain text:" label followed by the plaintext characters.
2. Write the "Key:" label followed by the key characters.
3. Write the "Encrypted text:" label followed by the encrypted characters.
4. Write the "Decrypted text:" label followed by the decrypted characters.

I add newline characters between each section to format the output clearly, matching the example exactly.

6. Handling Special Characters in Encrypted Output

I note that XORing uppercase and lowercase letters can result in non-alphanumeric ASCII characters, including spaces (decimal 32). The instructions confirm this is expected (e.g., 'L' XOR 'l' equals 32). This ensures the encrypted output remains consistent with the project's example.

7. Final Verification and Compliance

I verify the decrypted text exactly matches the original plaintext, showing that encryption and decryption are correctly implemented. The output file matches the required format and contains all the correct labels and data.