COAL – Lab: Project Report

Member Names	Member ID#
Huzefa Saifuddin	22K-5125
Muhammad Suhaib Qazi	22K-5073
Areeb-ur-Rehman	22K-6003

Project Report: Creating and Reading a File (assembly_code.asm)

Introduction

This project aims to demonstrate the fundamental concepts of file I/O operations in assembly language. The program, titled "CreateReadFile.asm," utilizes the Irvine32 library and macros to perform file creation, reading, and writing operations. It effectively handles user input, file error conditions, and displays relevant information to the console.

Program Structure

The program is divided into two main segments: the data segment and the code segment. The data segment defines various constants and variables, including the buffer size, filename, file handle, string length, bytes written, and three string buffers for error messages and user prompts. The code segment contains the main procedure, which orchestrates the entire program flow.

Program Functionality

1. **Create a New Text File:**

- The program opens the filename "output.txt" in write mode using the `CreateOutputFile` function.

- If the file creation fails, it displays an error message using the `WriteString` function and exits the program.

2. **User Input and String Processing:**

- The program prompts the user to enter a string using the `WriteString` function and the `ReadString` function.

- The entered string is stored in the buffer, and its length is stored in the `stringLength` variable.

3. **Write Buffer to Output File**:

- The program writes the contents of the buffer to the output file using the `WriteToFile` function.

- The number of bytes written is stored in the `bytesWritten` variable.

4. **Display Feedback and User Input for File Reading:**

- The program informs the user about the number of bytes written using the `WriteString` and `WriteDec` functions.

- It then prompts the user to enter an input filename using the `mWrite` function and the `ReadString` function.

5. **Open Input File and Check for Errors:**

- The program opens the specified input file in read mode using the `OpenInputFile` function.

- If the file opening fails, it displays an error message using the `mWrite` function and exits the program.

6. **Read File Contents and Display:**

- The program reads the contents of the input file into the buffer using the `ReadFile` function.

- It checks if the buffer size is sufficient to hold the entire file content using the `check_buffer_size` procedure.

- If the buffer is too small, it displays an error message using the `mWrite` function and exits the program.

- If the buffer is large enough, it inserts a null terminator at the end of the buffer and displays the buffer contents using the `mWrite` and `WriteString` functions.

7. **Close Files and Exit:**

- The program closes both the output file and input file using the `CloseFile` function.

- It then exits the program using the exit function.

Conclusion

The program successfully demonstrates the ability to create, read, and write text files in assembly language. It effectively handles user input, error conditions, and provides clear feedback to the user. The program serves as a valuable tool for understanding the fundamentals of file I/O operations in assembly language programming.