**COMPUTER SCIENCE DEPARTMENT**

Computer Organization and Assembly Language

**Report**

**Project: File Handeling and Encryption**

**Last date of Submission: 21th March 2024**

# Submitted To: Sir Ahmed Saleem Khattak

|  |  |
| --- | --- |
| **UBAID-BIN-WARIS** | **2212416** |
| **Maryam Shahzad** | **2212396** |
| **Muhammad Abdul Rehman** | **2212399** |
| **Wasfa Nauman Bhatti** | **2312462** |
| **Afaq Ahmed** | **2212367** |

Table of Contents

**Front Page01**

**Table of content02**

**Introduction03**

**Features04**

**Code Implementation05**

**Code06**

**Output08**

**Conclusion09**

**References10**

**Introduction**

In today's digital age, the security of sensitive information is of utmost importance. With the proliferation of digital communication and data storage, ensuring the confidentiality and integrity of data has become a critical concern for individuals and organizations alike. Encryption stands as a cornerstone in the realm of cybersecurity, offering a robust mechanism to protect data from unauthorized access and interception.

The project at hand focuses on the development of a simple file encryption and decryption program implemented in assembly language. This program serves as a practical tool for users seeking to safeguard their data stored in files through encryption, as well as to retrieve encrypted data through decryption. By providing a straightforward solution with basic encryption techniques, the program aims to address the need for lightweight yet effective data security measures.

In this report, we will delve into the functionalities, implementation details, and significance of the file encryption and decryption program, highlighting its importance in enhancing data security in today's digital landscape.

**Code Features**

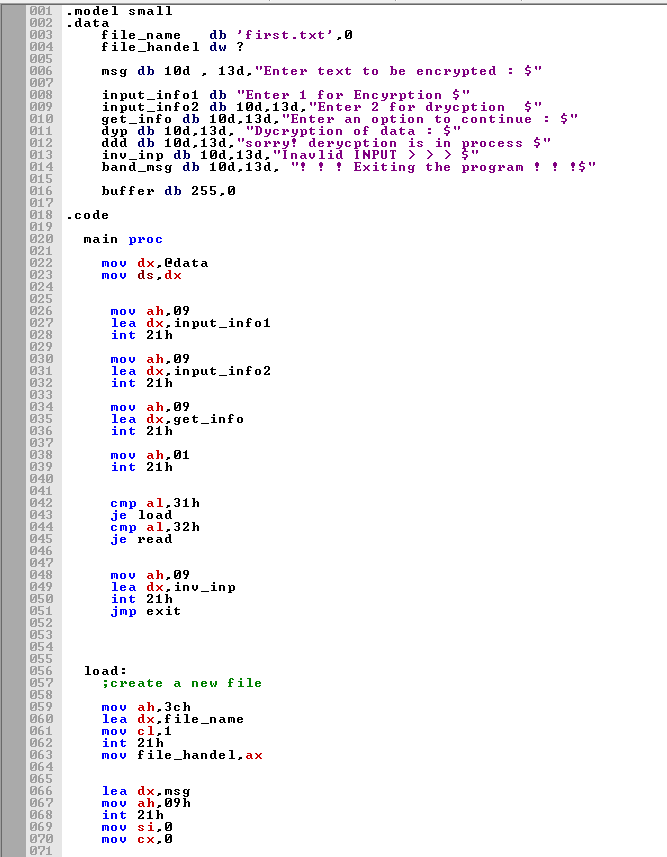
* **User-Friendly Interface:** The program offers a simple and intuitive user interface, guiding users through the encryption and decryption process with clear instructions and prompts.
* **Encryption and Decryption:** Users have the option to encrypt files to protect their contents or decrypt encrypted files to retrieve the original data. This dual functionality provides flexibility for users based on their security needs.
* **Basic Encryption Technique:** Employing a basic encryption technique, the program adds a fixed value (21 in this case) to the ASCII code of each character in the file. This straightforward approach ensures ease of implementation while providing a level of data security.
* **Error Handling:** The program includes error handling mechanisms to manage invalid inputs and unexpected scenarios gracefully. Users are guided with informative messages in case of incorrect inputs or file-related issues.
* **Customizable Encryption Key:** Although the program utilizes a fixed value for encryption, users can customize the encryption key by modifying the source code. This allows for some degree of customization and experimentation with different encryption strategies.
* **Support for Textual Data:** The program is designed to handle textual data, making it suitable for encrypting and decrypting text-based files such as documents, notes, and messages.
* **Lightweight Implementation:** Implemented in assembly language, the program offers a lightweight solution with minimal resource overhead. It can run efficiently on a wide range of systems without significant performance impact.
* **Compatibility:** The program is compatible with DOS-based systems, ensuring broad compatibility across legacy and modern computing environments. It can be executed on various platforms supporting DOS emulation or compatibility modes.

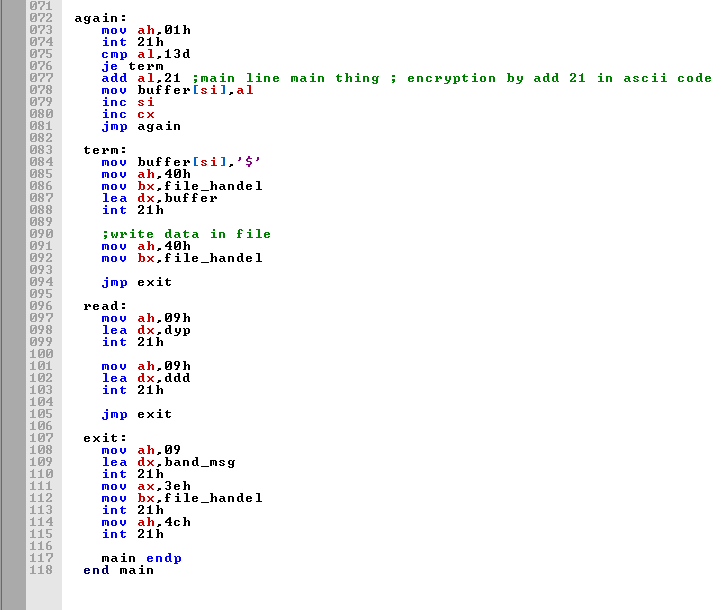
These features collectively contribute to the functionality and usability of the file encryption and decryption program, providing users with a practical tool for securing their sensitive data with minimal complexity and resource requirements

**Code Implementation**

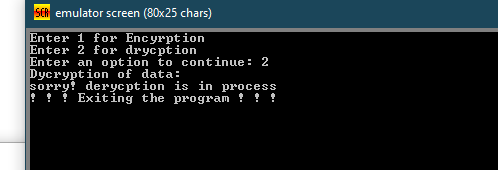
1. **Opening and Creating Files**
   * The program utilizes DOS interrupt int 21h to open existing files for decryption or create new files for encryption.
   * File names are stored as ASCII strings and passed to the appropriate interrupt functions along with the desired file operation parameters.
2. **Encryption and Decryption Logic**
   * Encryption is achieved by reading each byte from the input file, adding a fixed value (21h) to its ASCII code, and writing the modified byte to the output file.
   * Decryption follows a similar process but involves subtracting the fixed value (21h) from each byte's ASCII code.
3. **User Interface**
   * The program presents a simple text-based user interface with clear prompts and instructions.
   * Users are guided through the encryption/decryption process and prompted to enter their choice of action.
4. **Input Handling**
   * Input from the user, such as selecting encryption or decryption mode, is read using DOS interrupt int 21h with appropriate function codes.
   * Error handling mechanisms ensure that invalid inputs are detected and appropriate messages are displayed to guide the user.
5. **Buffer Management**
   * A buffer is allocated in memory to temporarily store data read from files during encryption/decryption.
   * Buffer management ensures efficient handling of data and minimizes memory usage.
6. **File Operations**
   * File read and write operations are performed using DOS interrupts int 21h, with proper file handles and buffer addresses specified.
   * Files are closed after encryption/decryption to release system resources and ensure data integrity.
7. **Flow Control**
   * The program logic utilizes conditional jumps (jmp) and loop structures to control the flow of execution based on user inputs and file processing requirements.
   * Proper termination procedures are implemented to gracefully exit the program after completing encryption/decryption tasks or in case of errors.
8. **Assembly Language Optimization**
   * Assembly language instructions are chosen carefully to optimize performance and minimize resource usage.
   * Efficient use of registers and memory ensures smooth execution on a wide range of systems, including legacy hardware configurations.
9. **Documentation and Comments**
   * The source code is thoroughly documented with comments to explain the purpose and functionality of each section.
   * Clear and concise comments help developers understand the codebase and facilitate future modifications or enhancements.

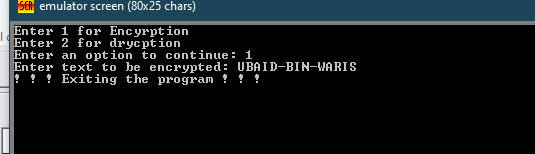
**Code**





**Output**







**Conclusion**

In conclusion, the file encryption and decryption program developed in assembly language provides a practical solution for securing sensitive data stored in files. By leveraging basic encryption techniques and a straightforward user interface, the program offers users a lightweight yet effective tool for enhancing data security.

The implementation of encryption and decryption logic, coupled with proper error handling and user guidance, ensures smooth execution and reliable performance. The program's compatibility with DOS-based systems and efficient resource utilization make it suitable for a wide range of computing environments.

While the program offers a simplified approach to file encryption and decryption, it serves as a stepping stone for understanding fundamental concepts in cybersecurity and assembly language programming. Future enhancements could include support for additional encryption algorithms, improved user interface elements, and integration with modern operating systems.

Overall, the file encryption and decryption program underscores the importance of data security in today's digital landscape and demonstrates the potential of assembly language programming in addressing real-world challenges. As digital threats continue to evolve, initiatives like this project play a crucial role in empowering users with the tools and knowledge to safeguard their valuable information.

**References**

1. [8086 Data Transfer Instructions - Assembly Language Programming (microcontrollerslab.com)](https://microcontrollerslab.com/8086-data-transfer-instructions-assembly-language-programming/)
2. [assembly-codes/2s\_complement.asm at main · amritoo/assembly-codes (github.com)](https://github.com/amritoo/assembly-codes/blob/main/2s_complement.asm)
3. [assembly-codes/condition.asm at main · amritoo/assembly-codes (github.com)](https://github.com/amritoo/assembly-codes/blob/main/condition.asm)
4. [13\_Assembly1.pdf (princeton.edu)](https://www.cs.princeton.edu/courses/archive/spr19/cos217/lectures/13_Assembly1.pdf)
5. [17ComputerArchitecture.pdf (princeton.edu)](https://www.cs.princeton.edu/courses/archive/spr06/cos217/lectures/17ComputerArchitecture.pdf)
6. [dos - What does it mean by "MOV AH, 4CH" in assembly language? - Stack Overflow](https://stackoverflow.com/questions/42573828/what-does-it-mean-by-mov-ah-4ch-in-assembly-language)