Report: File Operations Program

Student Name: [al amin Hossain nayem]

---

1. Introduction

This MASM assembly program implements a **menu-driven file operations system** for DOS. It allows users to create, edit, clean, rename files, change file dates, list directory contents, and print file content. The program uses **DOS interrupts (INT 21h)** for all input/output and file operations.

The program demonstrates the use of **small memory model**, keyboard input handling, and basic file manipulation techniques in assembly language.

---

2. Memory Model

- The program uses the **Small Memory Model**:
  - Single **64KB Code Segment**
  - Single **64KB Data Segment**
- Stack size: 256 bytes (.stack 100h)
- This memory model is suitable for small programs that fit entirely in one segment.

---

3. Program Structure

The program is **modularized into procedures** for each file operation. The main program displays a menu and calls the corresponding procedure based on user input.

3.1 Data Segment

- filename: Stores the original file name (your_name.txt)
- newfilename: Stores the new name for renaming (renamed.txt)
- menu_text: Menu options displayed to the user
- msg_*: Messages for success or error notifications
- kbd_buffer: DOS-style input buffer

- input_buffer: Buffer for reading file content
- file_handle: Stores file handles
- dta_buffer: Directory Table of Allocation buffer for directory listing
- day, month, year: Variables for file date manipulation

---

4. Algorithm and Logic

The program is a **menu-driven procedural program**. The **main loop** displays the menu, waits for user input, and executes the corresponding procedure.

4.1 Menu Options

1. **Create File**
   - Deletes the file if it exists
   - Creates a new empty file
   - Uses DOS interrupts: 41h (delete), 3Ch (create), 3Eh (close)
2. **Edit File**
   - Prompts the user for name and surname
   - Opens the file in read/write mode (3Dh)
   - Moves file pointer to the end (42h)
   - Writes user input to file (40h)
   - Adds newline
3. **Clean File**
   - Deletes the file (41h)
   - Recreates an empty file (3Ch)
4. **Rename File**
   - Renames file using DOS interrupt 56h
5. **Change File Date**
   - Prompts user to enter date (DD/MM/YYYY)

    - Converts input to **DOS date format**
    - Updates file date using interrupt 57h

6. **List Directory Files**
    - Uses DOS **Directory Table of Allocation (DTA)**
    - Find first file: 4Eh, Find next file: 4Fh
    - Prints filenames
7. **Print File Content**
    - Opens file (3Dh)
    - Reads content byte by byte (3Fh)
    - Prints each character to screen (02h)

---

5. Input/Output Handling

- **Keyboard input**
    - Single character: 01h
    - String input (DOS buffer): 0Ah
- **Console output**
    - Display string: 09h
    - Display single character: 02h
- **File I/O**
    - Create: 3Ch
    - Open: 3Dh
    - Close: 3Eh
    - Read: 3Fh
    - Write: 40h
    - Delete: 41h
    - Rename: 56h

    - Set date: 57h

---

6. Error Handling

- Every file operation checks the **Carry Flag (CF)** to detect errors.
- On error, a generic message "Error occurred!" is displayed.
- Example: jc create_error jumps to error handler if the create operation fails.

---

7. Helper Procedures

1. **read_two_digits**
    - Reads two numeric characters from user input
    - Converts them to decimal
2. **read_four_digits**
    - Reads four numeric characters
    - Converts to decimal
    - Converts to **DOS date format** (year since 1980)

---

8. Algorithmic Flow

1. **Initialize DS**
2. **Display Menu**
3. **Read User Option**
4. **Call Corresponding Procedure**
5. **Perform Operation (Create, Edit, Clean, etc.)**
6. **Return to Menu**
7. **Exit on Option 0**

Flowchart:

START

```
 │
 ▼
Display Menu
 │
 ▼
Read Option
 │
  ├─ Option 1 → Create File
  ├─ Option 2 → Edit File
  ├─ Option 3 → Clean File
  ├─ Option 4 → Rename File
  ├─ Option 5 → Change File Date
  ├─ Option 6 → List Directory
  ├─ Option 7 → Print File
 └─ Option 0 → Exit
 │
 ▼
Return to Menu (Loop)
```

---

9. Observations

- Fully **DOS-based**, interrupt-driven, synchronous program.
- Uses **low-level file handling** without C library or OS abstractions.
- Input validation is minimal; expects numeric format for dates.
- Modular approach makes the program **maintainable**.

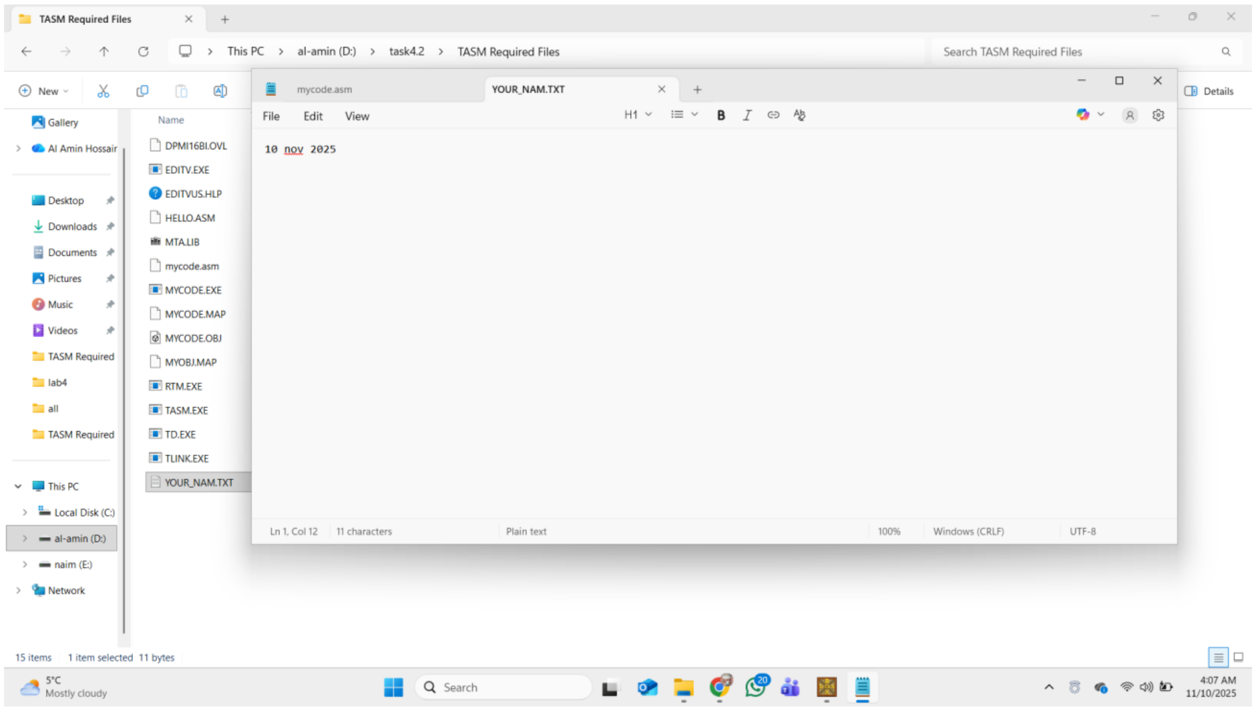
---

10. Conclusion

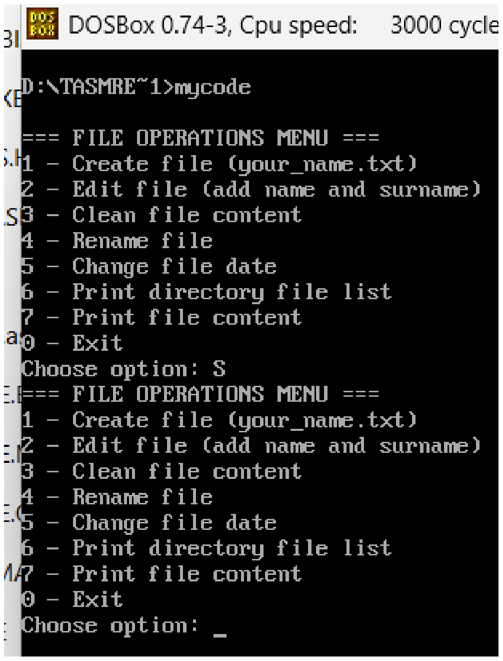
This program demonstrates:

- **File management in assembly** using DOS interrupts.
- **Procedural programming** and **modular design** in MASM.
- **Keyboard I/O and screen output handling**.
- **Directory listing, file renaming, and date manipulation**.

It can serve as a template for learning **low-level file operations**, **menu-driven programs**, and **assembly-based I/O**.

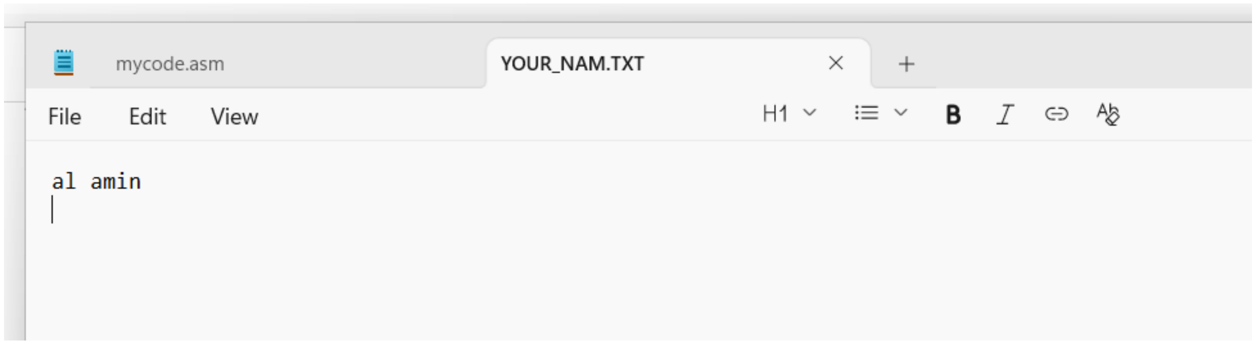


```
10 nov 2025
```

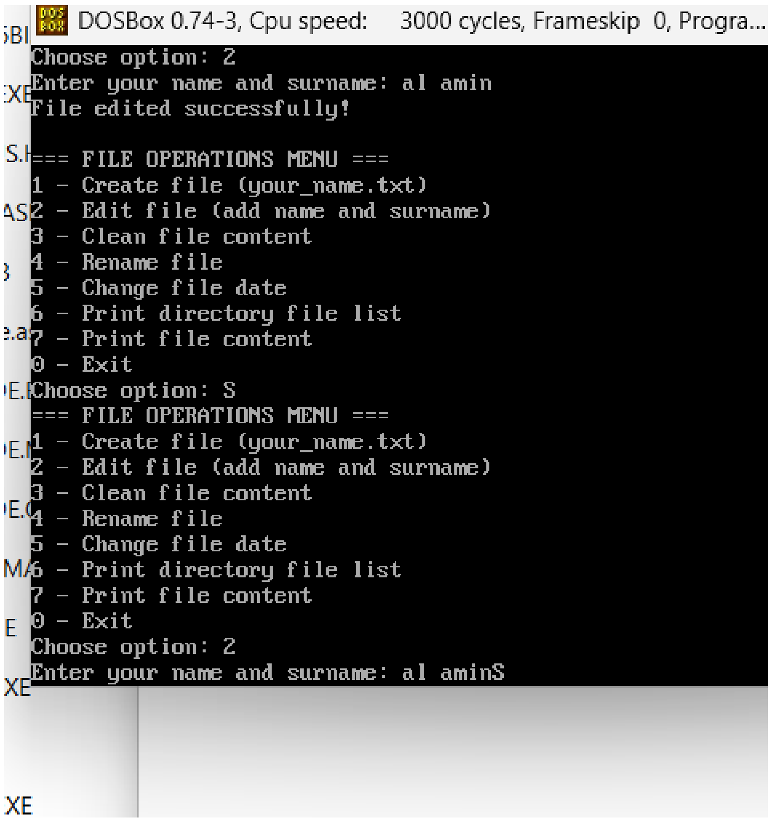


```
D:\TASMRE~1>mycode

=== FILE OPERATIONS MENU ===
1 - Create file (your_name.txt)
2 - Edit file (add name and surname)
3 - Clean file content
4 - Rename file
5 - Change file date
6 - Print directory file list
7 - Print file content
0 - Exit
Choose option: 5
=== FILE OPERATIONS MENU ===
1 - Create file (your_name.txt)
2 - Edit file (add name and surname)
3 - Clean file content
4 - Rename file
5 - Change file date
6 - Print directory file list
7 - Print file content
0 - Exit
Choose option: _
```



```
al amin
```



DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
Choose option: 2
Enter your name and surname: al amin
File edited successfully!

=== FILE OPERATIONS MENU ===
1 - Create file (your_name.txt)
2 - Edit file (add name and surname)
3 - Clean file content
4 - Rename file
5 - Change file date
6 - Print directory file list
7 - Print file content
0 - Exit
Choose option: S
=== FILE OPERATIONS MENU ===
1 - Create file (your_name.txt)
2 - Edit file (add name and surname)
3 - Clean file content
4 - Rename file
5 - Change file date
6 - Print directory file list
7 - Print file content
0 - Exit
Choose option: 2
Enter your name and surname: al aminS
```