**COMPUTER SCIENCE DEPARTMENT**

Computer Organization and Assembly Language (LAB)

**Report**

**Project: Encryption and Calculator**

**Last date of Submission: 15th May 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**

**Project overview03**

**Summary04**

**Code Feature04**

**Code Diagram06**

**Code07**

**Output12**

**Conclusion14**

**References14**

**Project Overview**

* **Data Section**
  + Defines various strings used for menu options, arithmetic operations, user prompts, and result displays.
  + Allocates memory for storing input strings and processed data.
* **Code Section**
  + Starts with the main program entry point labeled as menu.
  + Displays a menu with options for arithmetic operations, encryption/decryption, viewing group members, and exiting the program.
  + Utilizes interrupts for displaying strings and handling user input.
  + Processes user input using conditional jumps to navigate to specific functionalities.
  + Implements functionalities for arithmetic operations (addition, subtraction, multiplication, division), multiplication/division by 2, encryption/decryption of strings, and viewing group member information.
  + Uses loops to iterate through input strings for encryption/decryption.
  + Displays results and messages after executing each operation.
  + Allows the user to exit the program gracefully with a farewell message.
* **Features**
  + User-friendly menu interface.
  + Input validation and error handling for invalid inputs.
  + Arithmetic operations support.
  + Encryption/decryption functionality.
  + Displaying group member information.
  + Proper program termination.
* **Execution Flow**
  + The program starts with displaying the main menu.
  + It waits for user input and branches to the corresponding functionality based on the user's choice.
  + After executing the selected operation, it returns to the main menu unless the user chooses to exit.
  + Proper message prompts guide the user throughout the interaction.
* **Overall**
  + The code demonstrates basic assembly language programming concepts such as data declaration, conditional branching, looping, interrupt-based I/O operations, and user interaction.
  + It provides a structured and functional menu-driven interface for performing various tasks efficiently in a small model environment.

**Summary**

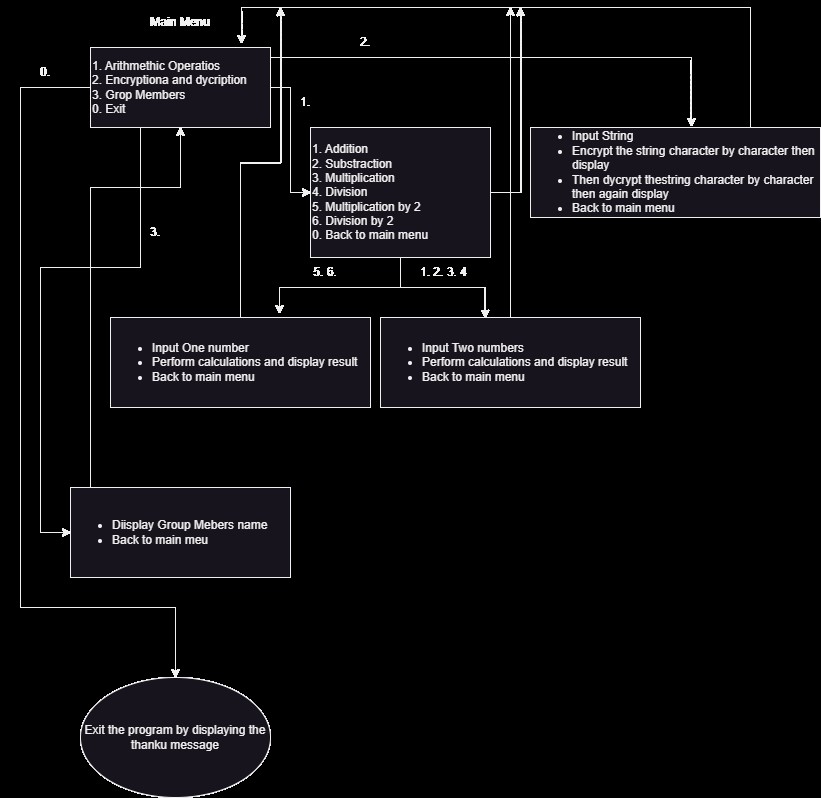
The assembly program is a menu-driven interface allowing users to perform arithmetic operations, encrypt/decrypt strings, view group member information, and exit the program. It displays a menu with numbered options for each functionality, prompts users for input, handles invalid inputs, and displays results accordingly. The program utilizes loops, conditional branching, and interrupts for input/output operations to provide a user-friendly experience.

**Code Features**

* **Menu Display**: The program displays a menu with options for arithmetic operations, encryption/decryption, viewing group members, and exiting the program.
* **Arithmetic Operations**
  + Addition
  + Subtraction
  + Multiplication
  + Division
  + Multiplication by 2
  + Division by 2
* **Encryption and Decryption**
* Users can input a string, which is then encrypted by performing a simple XOR operation with 'a' and displayed.
* The original string can be decrypted by performing the XOR operation again with 'a'.
* **Viewing Group Members**: Displays a list of group members with their IDs.
* **Input Handling**
  + Checks user input against menu options and executes corresponding functionalities.
  + Displays an error message for invalid inputs.
* **Output Display**
* Displays results of arithmetic operations (sum, difference, product, quotient, remainder).
* Displays the encrypted and decrypted strings.
* Displays a "thank you" message upon program exit.
* **Looping and Jumping**
* Uses loops and conditional jumps to navigate through menu options and perform operations.
* Loops through input strings for encryption/decryption.
* **User Interaction**
* Waits for user input at various stages to proceed.
* Prompts the user to press any key to continue after displaying certain messages.
* **Memory Allocation and Data Declaration**
* Declares variables to store strings, menu options, arithmetic results, and user inputs.
* Allocates memory for storing data and strings.
* **Program Termination**
* Allows the user to exit the program gracefully.
* Displays a farewell message before terminating the program.

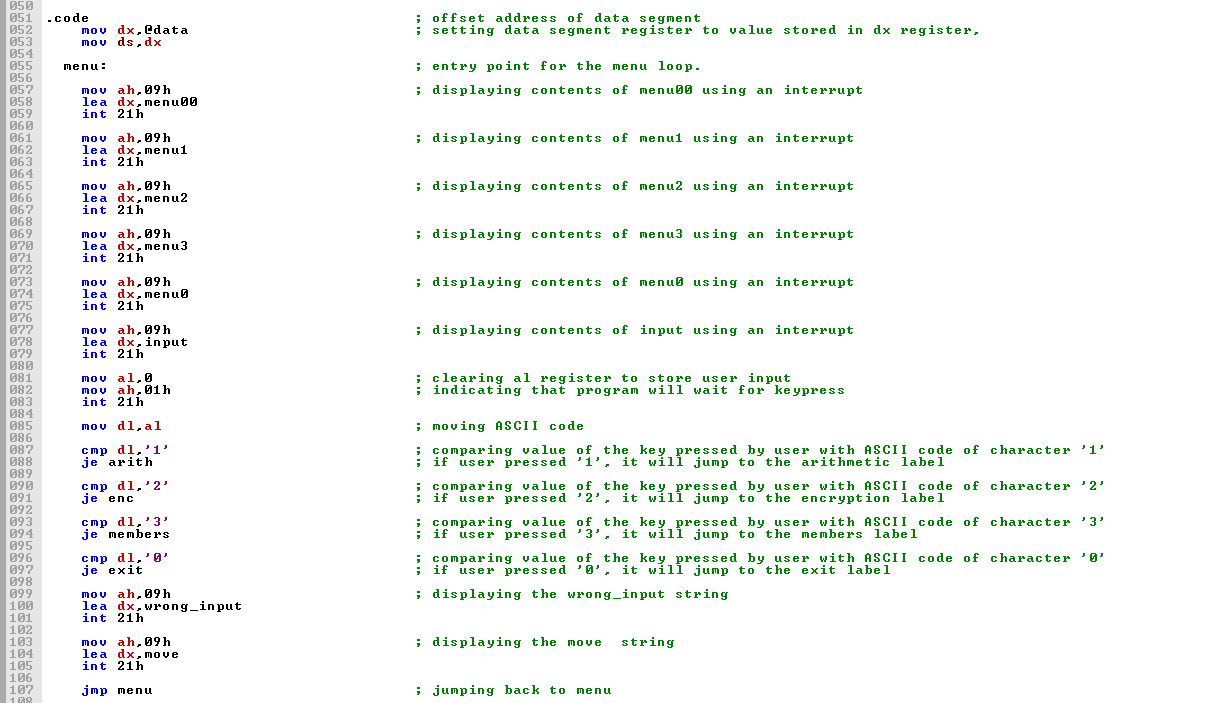
These features combine to create a simple but functional menu-driven assembly program capable of performing arithmetic operations, encryption/decryption, and displaying group member information.

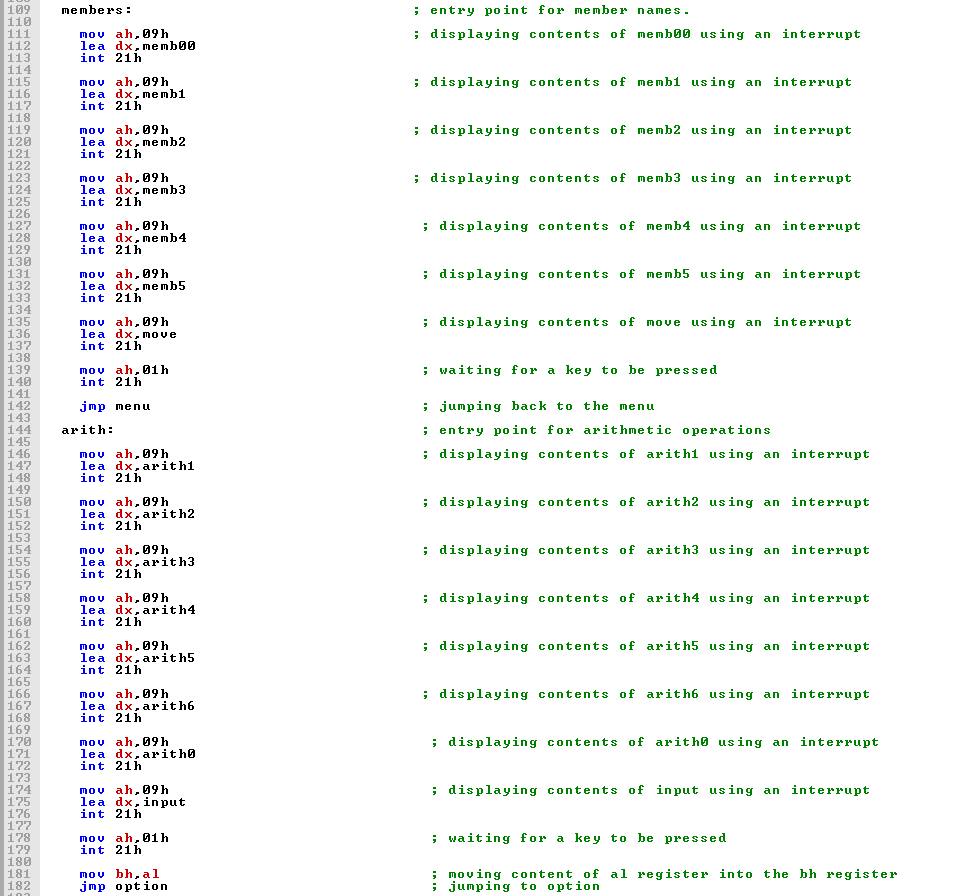
**Code Diagram**

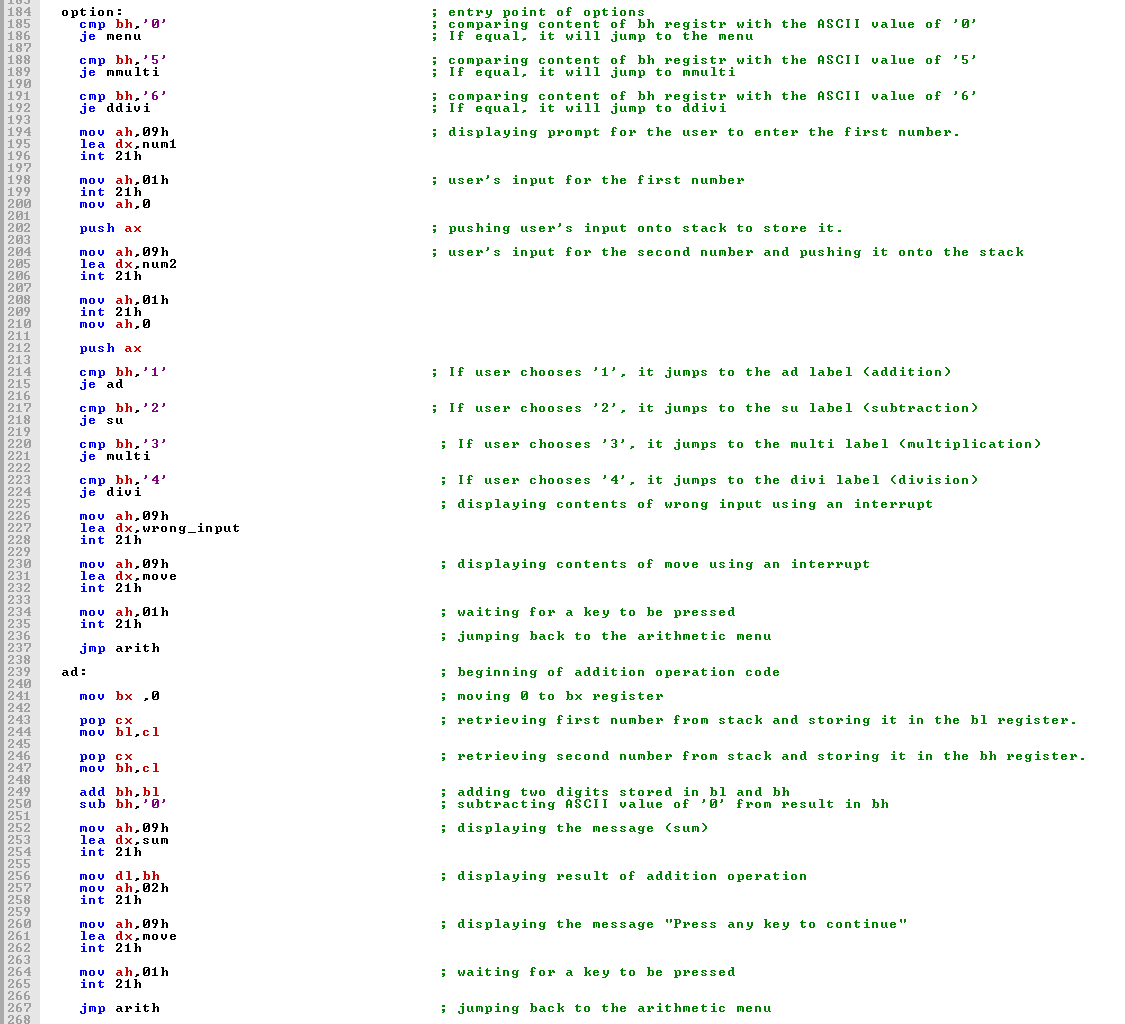


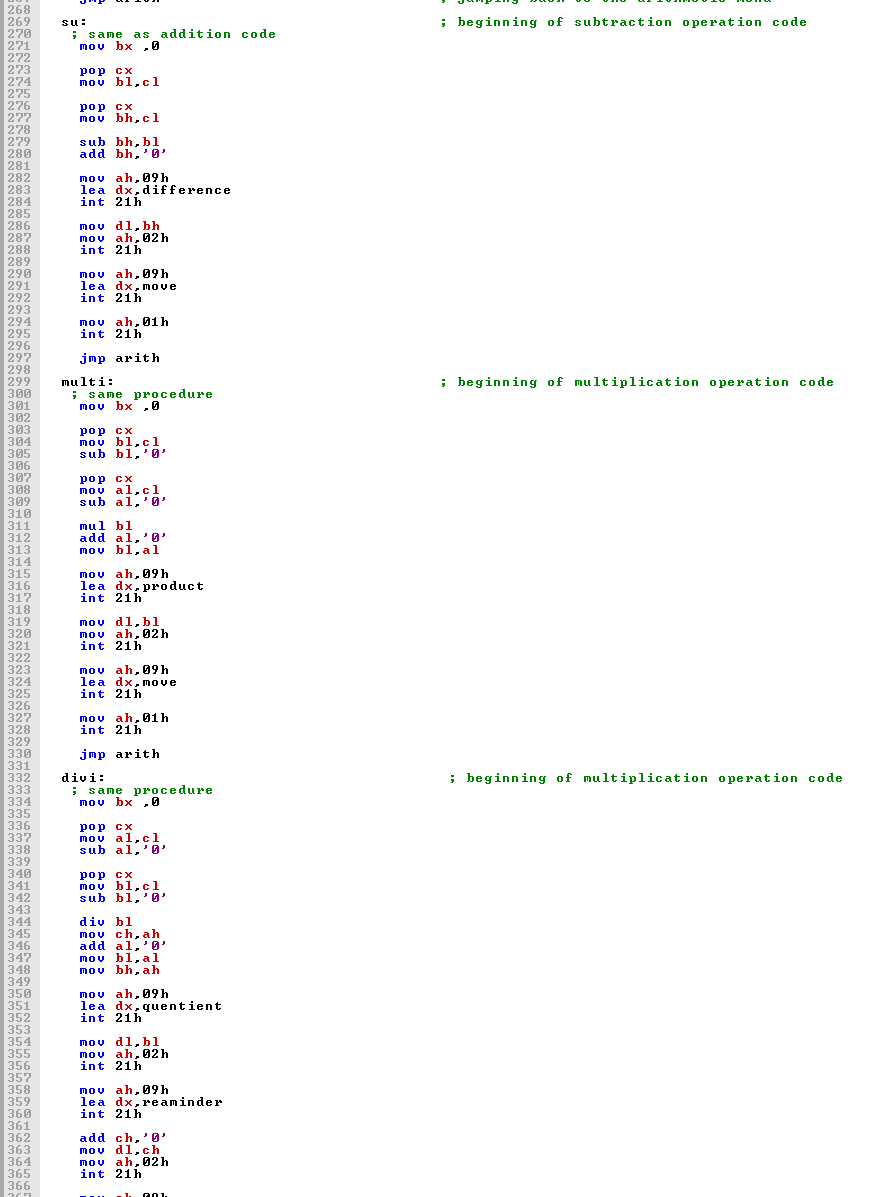
**Code**



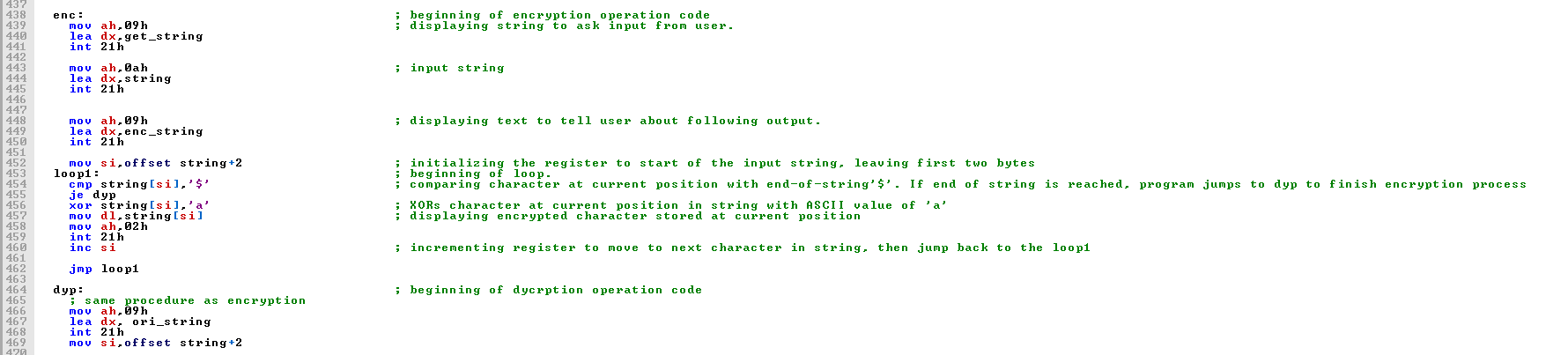


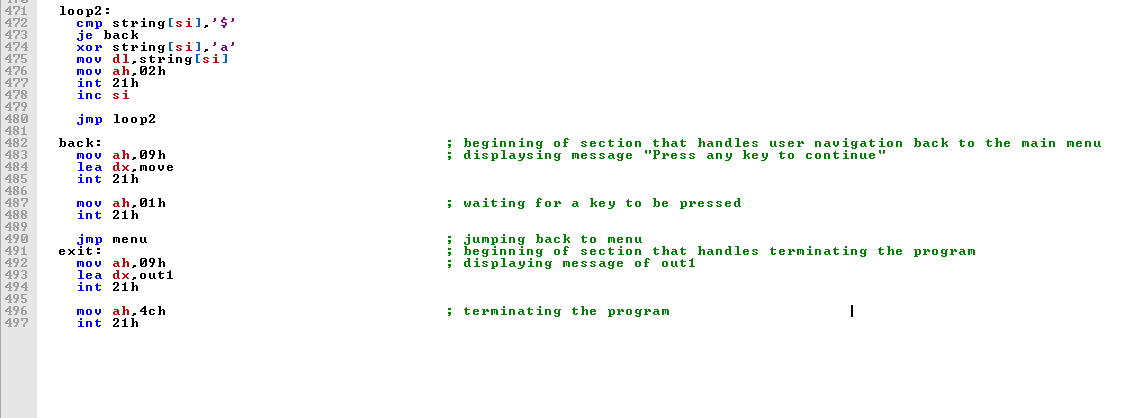




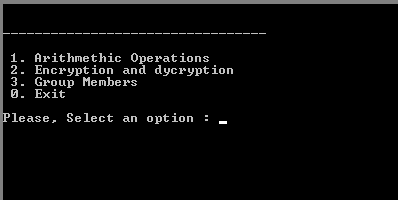


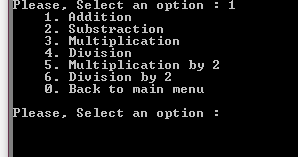


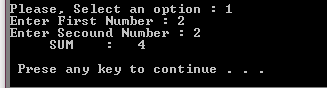


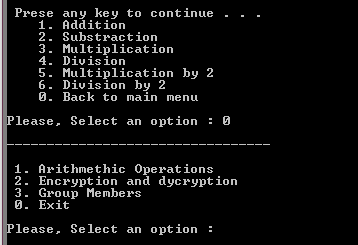


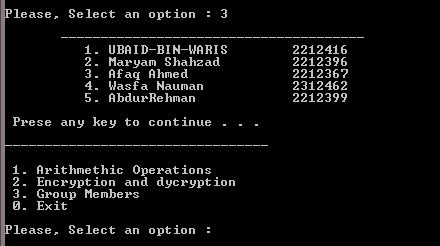
**Output**

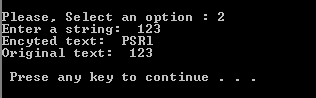














**Conclusion**

In conclusion, the assembly code presents a compact yet functional menu-driven program capable of performing arithmetic operations, encryption/decryption of strings, and displaying group member information. It leverages basic assembly language constructs such as conditional branching, loops, and interrupt-based input/output operations to provide a user-friendly interface. Despite its simplicity, the code demonstrates effective handling of user inputs, validation of choices, and proper execution of selected functionalities. Overall, it serves as a practical example of implementing interactive programs in assembly language within a limited memory model.

**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)