**assembly project**

**calculator**

### **1. Project Overview**

The calculator project is an assembly language program designed for the 8086 microprocessor. It performs basic arithmetic operations (addition, subtraction, multiplication, and division) and logical operations (AND, OR, XOR). The program uses user inputs to execute selected operations and displays the results interacti

### **2. Setup and Requirements**

#### **Tools Needed**

1. **8086 Emulator**: The program is designed for an 8086-compatible environment, such as EMU8086 or MASM.
2. **Assembler**: A tool to assemble the source code into machine code (e.g., TASM, MASM).
3. **Debugger** (optional): For step-by-step analysis and debugging.

#### **Steps to Compile and Execute**

1. Load the source file (final\_project.asm) into the emulator or assembler.
2. Assemble the program to generate the executable.
3. Run the assembled program in the 8086 emulator.
4. Follow the on-screen prompts to interact with the program.

### **3. Features and Functionality**

The calculator supports the following operations:

1. **Basic Operations**:
   * **Addition**
   * **Subtraction**
   * **Multiplication**
   * **Division**
2. **Logical Operations**:
   * **AND**
   * **OR**
   * **XOR**
3. **Program Exit**:
   * An option to terminate the program gracefully.



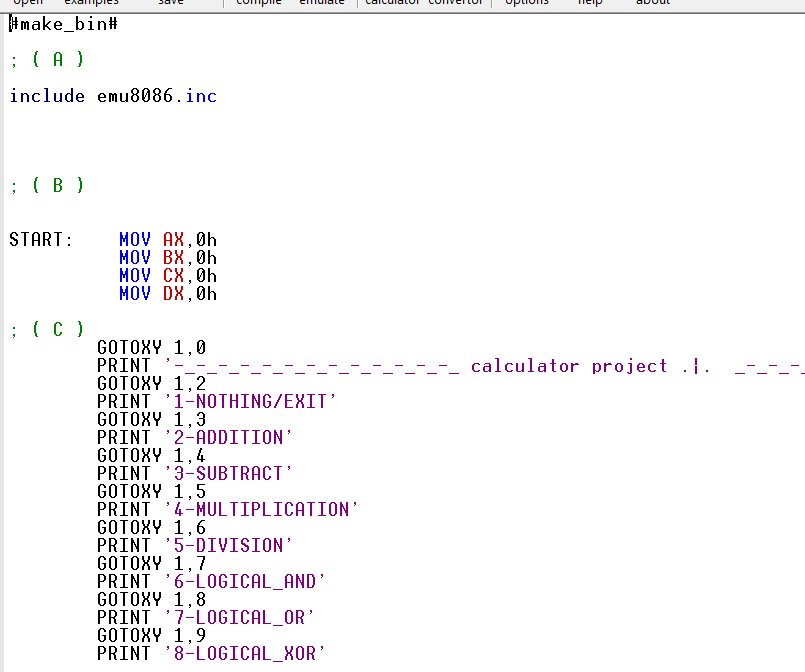
### **4. Code Structure**

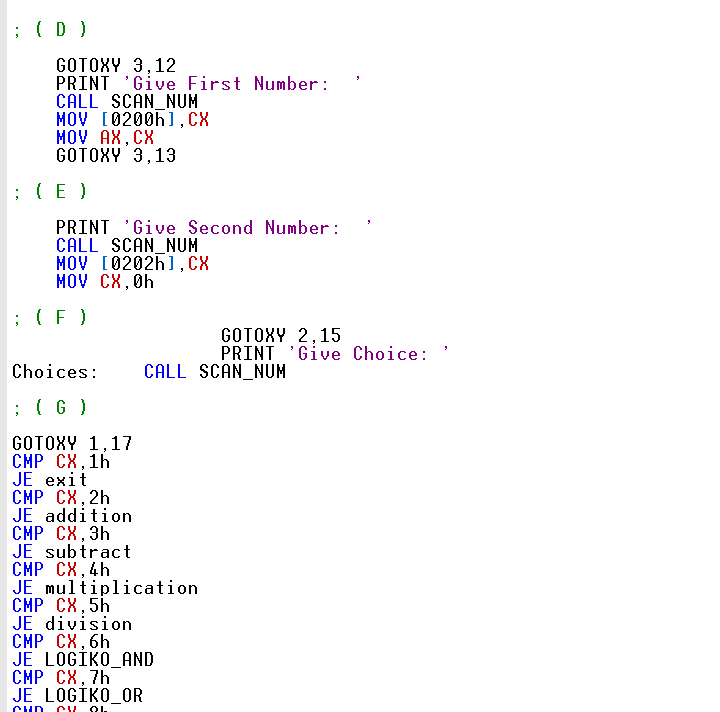
#### **main sections**

1. **initialization**
   * Registers (AX, BX, CX, DX) are cleared.
   * Memory locations are prepared for input and output storage.
2. **Menu Display**:
   * Prints the available operations for the user.
   * Displays prompts for input.
3. **Operation Handling**:
   * Based on user selection, the program jumps to the corresponding routine to perform the operation.
4. **R*esult Display****\*\*:*
   * *Outputs the result of the operation on the screen.*

#### **Key Variables and Memory Usage**

* CX: Used to store user input.
* [0200h]: Temporary storage for the first number.
* [0202h]: Temporary storage for the second number.
* Registers (AX, BX, etc.): Used for intermediate calculations.



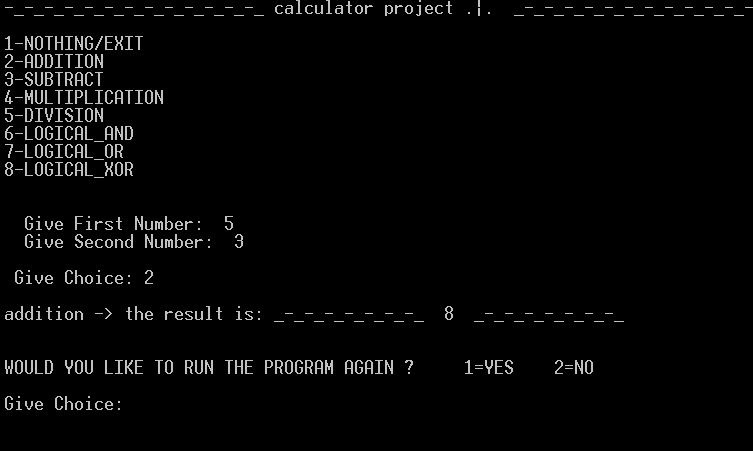


### **5. Instructions for Use**

1. Run the program in the emulator.
2. A menu will appear displaying the available operations.
3. Select an operation by entering the corresponding number (e.g., 2 for addition).
4. Input the first number when prompted.
5. Input the second number when prompted.
6. The result of the operation will be displayed on the screen.
7. To exit, select the "Exit" option from the menu.

#### **Example Interaction**

* User selects "2" for addition.
* Inputs 5 as the first number and 3 as the second number.
* The program displays: Result: 8.



### **6. Design and Implementation**

#### **Assembly Instructions Used**

* **mov**: To move data between registers and memory.
* **ADD, SUB, MUL, DIV**: Arithmetic operations.
* **AND, OR, XOR**: Logical operations.
* **CALL**: To invoke subroutines for specific tasks (e.g., scanning input, printing output).
* **GOTOXY, PRINT**: For screen manipulation and output display.

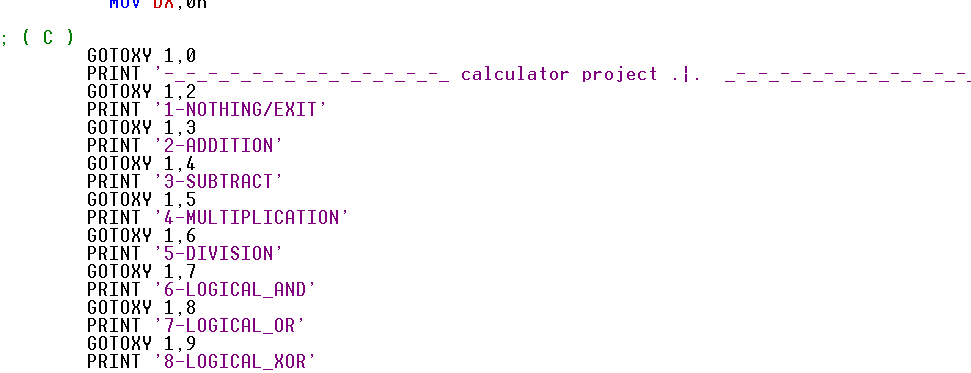
#### **Program Flow**

1. Display the menu.
2. Read user input for the operation.
3. Execute the corresponding subroutine.
4. Display the result.
5. Return to the menu or exit.

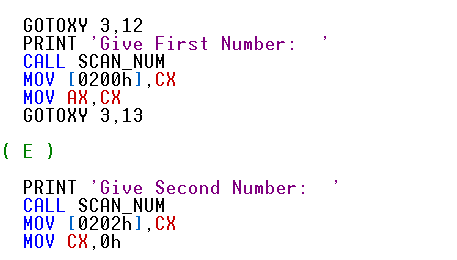
### **7. Appendix**

### **Code Snippets**

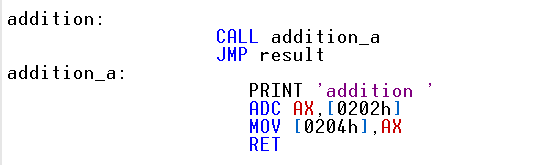
**Menu Display Section:**

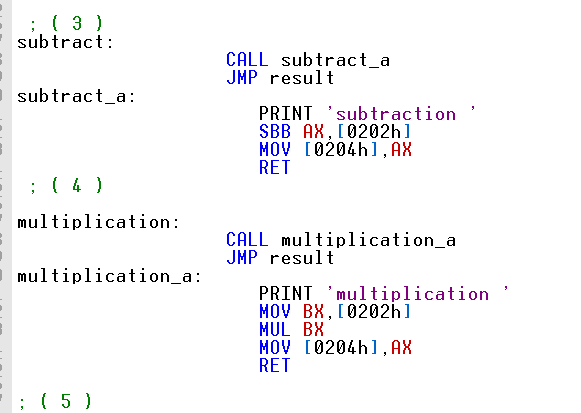
****

**Input Handling:**



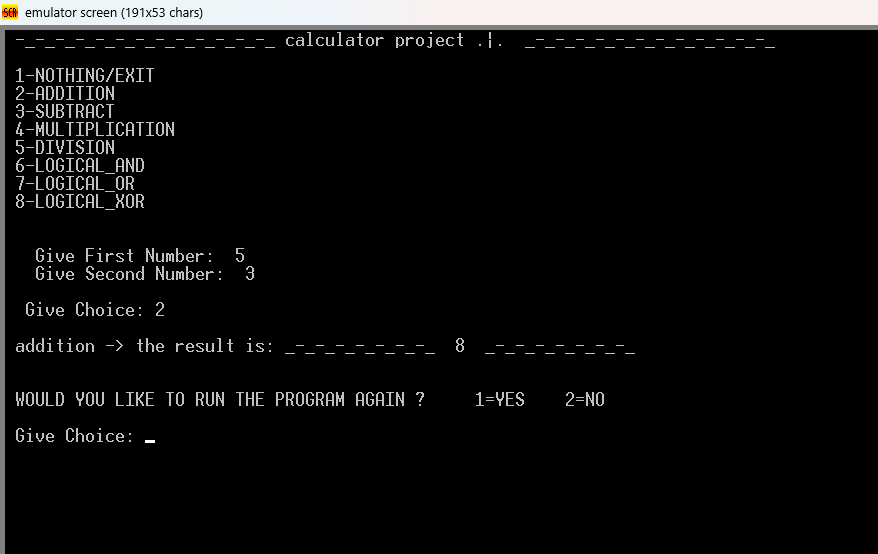
**Arithmetic Operations Example (Addition):**



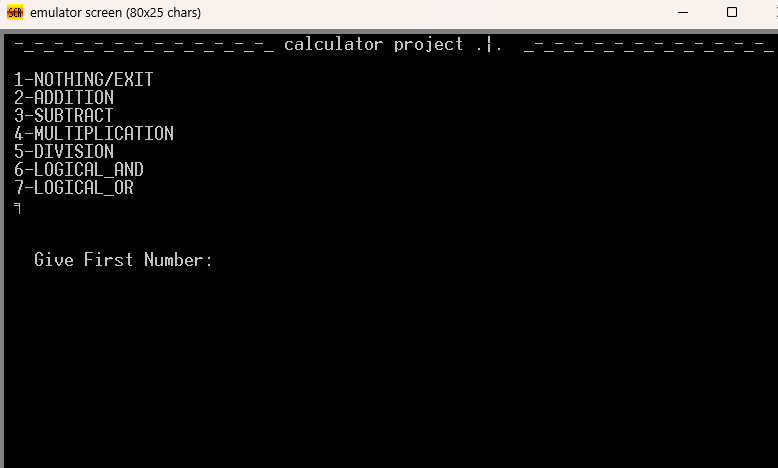


8.test the program

1.test some Arithmetic operations



2.test to continue in program



3.exit the program

