Logo, company name

Description automatically generated



**COMSATS University Islamabad (CUI)**

**Lab terminal**

**Submitted to:**

**Submitted By: ANIS MAJID**

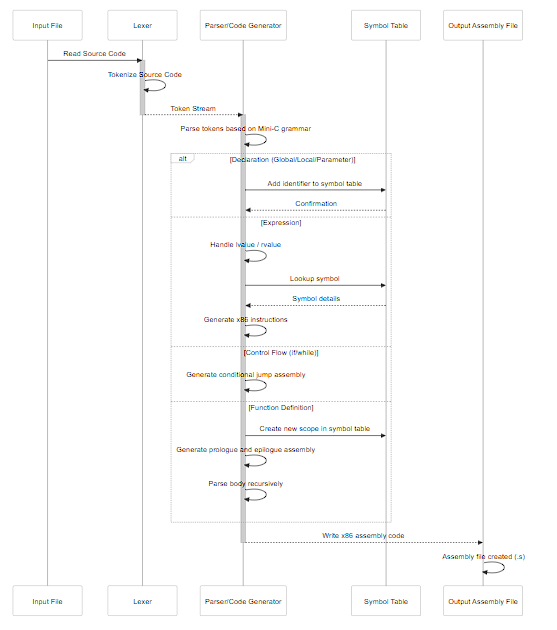
**Reg No: FA21-BCS-024**

**Course: COMPILER CONSTRUCTION**

**Date: 3 JANUARY 2025**

**Question No 01)**

**SEQUENCE DIAGRAM**

****

**Explanation of the diagram:**

* **Participants:**
  + **Input File:** Represents the source code file in the mini-c language.
  + **Lexer:** Responsible for reading the source code and breaking it into tokens.
  + **Parser/Code Generator:** Responsible for parsing the token stream, checking grammar rules, and generating x86 assembly code.
  + **Symbol Table:** Manages information about variables and functions, including their names and storage locations.
  + **Output Assembly File:** Represents the file where the generated assembly code is written.
* **Messages:**
  + **Read Source Code:** The Input File provides the source code to the Lexer.
  + **Tokenize Source Code:** The Lexer processes the source code, generating tokens.
  + **Token Stream:** The Lexer passes the stream of tokens to the Parser/Code Generator.
  + **Parse tokens based on Mini-C grammar:** The Parser/Code Generator analyzes the token stream based on the defined grammar.
  + **Add identifier to symbol table / Lookup symbol**: The Parser/Code Generator interacts with the Symbol Table, to store declarations and access info.
  + **Generate x86 instructions:** The Parser/Code Generator generates assembly instructions based on the parsed code.
  + **Handle lvalue / rvalue**: The Parser/Code Generator determines if an expression is an lvalue or a rvalue for the code generation.
  + **Generate conditional jump assembly**: The Parser/Code Generator creates assembly jumps for conditional code.
  + **Create new scope in symbol table**: The Parser/Code Generator initializes a new scope for each function.
  + **Generate prologue and epilogue assembly**: The Parser/Code Generator creates the function start and end instructions.
  + **Parse body recursively**: The Parser/Code Generator recursivelly parse the function code.
  + **Write x86 assembly code:** The Parser/Code Generator writes the generated assembly code into the Output Assembly File.
* **Activations:**
  + The Lexer and Parser/Code Generator participants are activated during their respective phases of processing.
* **Alternatives:**
  + The alt block highlights that the Parser/Code Generator handles different syntactic elements differently: declarations, expressions, control flow structures and function definitions.

This sequence diagram provides a high-level overview of the key steps and interactions within the mini-c compiler. It illustrates how different parts of the compiler work together to convert mini-c source code into x86 assembly code.