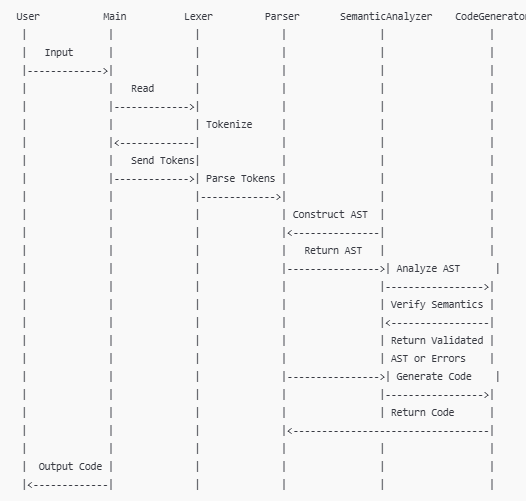
**Q5 : Draw a sequence diagram of MiniC Compiler :**

This is the general sequence diagram of miniC compiler :



**Explanation**

1. **User to Main**: User provides the Mini-C source code as input.
2. **Main to Lexer**: Main sends the source code to the Lexer for tokenization.
3. **Lexer to Main**: Lexer returns the tokens.
4. **Main to Parser**: Main sends the tokens to the Parser.
5. **Parser to Main**: Parser constructs the AST and returns it.
6. **Main to SemanticAnalyzer**: Main sends the AST for semantic analysis.
7. **SemanticAnalyzer to Main**: Semantic Analyzer validates the AST and returns either the validated AST or error information.
8. **Main to CodeGenerator**: Main sends the validated AST to the Code Generator.
9. **CodeGenerator to Main**: Code Generator returns the generated code.
10. **Main to User**: Main outputs the final compiled code or an error report.

**Specific Sequence Diagram of miniC:**



**Components (Lifelines):**

* **User:** Represents the individual who interacts with the compiler, providing input source code and receiving the final executable.
* **Compiler (minicc):** The core compiler program that orchestrates the entire process, coordinating the different compiler stages.
* **Lexer (lexer.mll):** The lexical analyzer, responsible for breaking the source code down into tokens.
* **Parser (parser.mly):** The syntax analyzer, which groups the tokens into structured Abstract Syntax Tree (AST) representations.
* **Semantic (semantic.ml):** The semantic analyzer, used to validate the meaning of the code, by checking the types, scopes and performing declaration checks.
* **CodeGen (codegen.ml):** The code generator, responsible for translating the AST into x86-64 assembly code.
* **Assembler (gcc):** The external assembler and linker (such as gcc), that combines the assembly code and libraries, to produce the final executable program.