Mini Compiler Sequence Diagram

```
+--------+          +-----------+           +--------+          +--------+
+-----------+          +--------+           +-----------+          +---------+
| User  |           | Compiler |            | Lexer |           |Parser |
|Semantic  |           |Intermed|            | Optimizer|           |Output  |
+--------+          +-----------+           +--------+          +--------+
+-----------+          +--------+           +-----------+          +---------+
    |                      |                    |                     |
|                      |                     |                      |
    | Inputs source code    |                    |                     |
|                      |                     |                      |
    +---------------------->|                    |                     |
|                      |                     |                      |
                             | Tokenize source   |                     |
|                      |                     |                      |
                             | code              |                     |
|                      |                     |                      |
                             +------------------>+                     |
|                      |                     |                      |
                             |                  | Return tokens        |
|                      |                     |                      |
                             +<-----------------+                     |
|                      |                     |                      |
                             |                  | Check syntax          |
|                      |                     |                      |
                             +----------------------------------------->+
|                      |                     |                      |
                             |                  | Parse tree or error   |
|                      |                     |                      |
                             +<-----------------------------------------+
|                      |                     |                      |
                             |                  | Validate semantics     |
|                      |                     |                      |
                             +---------------------------------------------------
--------->+                      |                     |                      |
                             |                  | Semantic validation     |
|                      |                     |                      |
                             +<--------------------------------------------------
----------+                      |                     |                      |
                             |                  | Generate intermediate code
|                      |                     |                      |
                             +---------------------------------------------------
----------------------->+          |                     |                      |
                             |                  | Intermediate code
|                      |                     |                      |
                             +<--------------------------------------------------
------------------------+          |                     |                      |
                             |                  | Optimize intermediate code
|                      |                     |                      |
                             +---------------------------------------------------
---------------------------------------------------->+      |
|
                            |                  | Optimized code
|                     |                     |                     |
                            +<-----------------------------------------------
----------------------------------------------------+     |
|
                            |                  | Generate target code
|                     |                     |                     |
                            +------------------------------------------------
----------------------------------------------------------------------------->+
|
                            |                  | Machine/executable code
|                     |                     |                     |
                            +<-----------------------------------------------
------------------------------------------------------------------------------+
|
                            |                  | Produce final output
|                     |                     |                     |
                            +------------------------------------------------
--------------------------------------------------------------------------------
------->+
```

Actors and Components:

1. **User** (inputting code)
2. **Lexical Analyzer (Lexer)**: Tokenizes the input.
3. **Syntax Analyzer (Parser)**: Checks grammar/syntax.
4. **Semantic Analyzer**: Validates semantic rules.
5. **Intermediate Code Generator**: Produces intermediate representation.
6. **Code Optimizer**: Optimizes the intermediate code.
7. **Code Generator**: Produces target code (e.g., assembly/machine code).
8. **Output** (final executable file or error report)

Sequence Description:

1. **User** inputs the source code.
   - Flow: User → Compiler
2. The **Lexical Analyzer** tokenizes the source code into tokens (keywords, operators, identifiers, etc.).
   - Flow: Compiler → Lexer → Compiler
3. The **Syntax Analyzer** checks the token sequence for grammar correctness based on predefined rules.
   - Flow: Compiler → Parser → Compiler

4. The **Semantic Analyzer** ensures the code adheres to semantic rules (e.g., type checking, scope rules).
    - Flow: Compiler → Semantic Analyzer → Compiler
5. The **Intermediate Code Generator** produces intermediate code (e.g., three-address code).
    - Flow: Compiler → Intermediate Code Generator → Compiler
6. The **Code Optimizer** refines intermediate code for performance (e.g., removes redundant instructions).
    - Flow: Compiler → Code Optimizer → Compiler
7. The **Code Generator** converts optimized code into machine code or assembly.
    - Flow: Compiler → Code Generator → Compiler
8. The compiler produces the **Output** (executable file or error report).
    - Flow: Compiler → Output