# COMPILER CONSTRUCTION LAB TERMINAL


# Muhammad Hammad Nazir
# Fa21-Bcs-048


**QUESTION NO.3**

Describe the Optimization in the Mini Compiler:

---

1. Source Code Analysis:

The process begins by analyzing the source code written in a high-level programming language. The goal is to identify areas for improvement while preserving the program's functionality.

2. Tokenization:

The source code is tokenized into smaller units, such as keywords, operators, identifiers, and literals. These tokens help identify redundant or inefficient constructs in the code.

3. Semantic Analysis and Error Detection:

Semantic analysis ensures the correctness of the code's meaning by performing type checking, symbol table creation, and adherence to language semantics. Errors or inefficiencies detected at this stage are flagged for correction.

4. Intermediate Representation (IR) Generation:

The code is transformed into an intermediate representation (IR), a simplified and standardized form that allows for optimization techniques to be applied across different programming languages.

5. Optimization Techniques:

- Code Elimination: Removing redundant or unreachable code.

- Loop Optimization: Enhancing loop execution, such as unrolling or invariant code motion.

- Inline Function Expansion: Replacing function calls with their definitions to reduce overhead.

- Constant Folding: Computing constant expressions at compile time.

- Dead Code Removal: Removing code that does not affect program output.

6. Memory and Resource Optimization:

Memory is allocated efficiently for variables and data structures, with dynamic allocation and deallocation as needed. Optimizations include minimizing memory usage and improving cache performance.

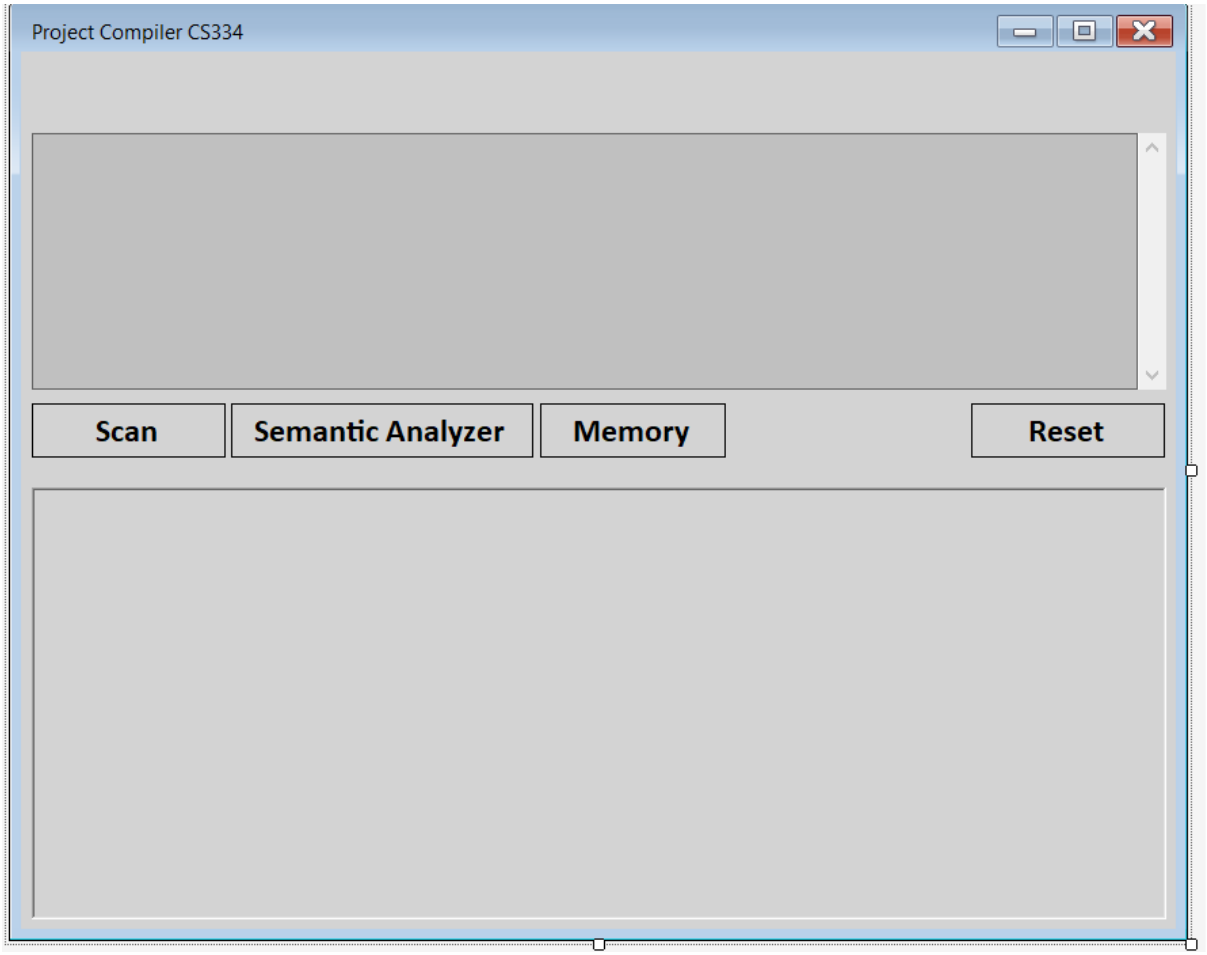
7. Error Reporting:

Any errors or warnings detected during semantic analysis, memory management, or optimization are reported to the programmer for resolution.

8. Final Code Generation:

The optimized IR is converted into machine code, ensuring it runs efficiently on the target platform without sacrificing correctness.

## Screenshots:


Project Compiler CS334
Scan
Semantic Analyzer
Memory
Reset

Project Compiler CS334