**Q1 . Brief of the project**

• Lexical Analyzer (Scanner):

This component breaks the source code into a sequence of tokens by analyzing the characters.

• Syntax Analyzer (Parser):

It checks the syntax of the source code based on the grammar rules of the programming language and generates a syntax tree.

• Semantic Analyzer:

This component ensures that the semantics of the source code are correct. It checks for meaningful constructs and relationships between symbols.

• Intermediate Code Generator:

Converts the source code into an intermediate representation, making it easier to optimize and target different platforms.

• Code Optimizer:

Enhances the intermediate code to improve the program's performance. It includes techniques like constant folding, loop optimization, and dead code elimination.

• Code Generator:

Translates the optimized intermediate code into machine code or another high-level language, depending on the target platform.

• Symbol Table:

Manages information about symbols (variables, functions, etc.) in the source code, facilitating scope resolution and type checking.

• Error Handler:

Detects and reports errors in the source code, providing meaningful error messages to assist developers in debugging.

• Run-time Library:

A set of pre-written code that handles certain operations at runtime, such as input/output, memory management, and other language-specific features.

• Loader and Linker:

If the compiler generates object code, a loader loads the code into memory for execution, and a linker combines it with other necessary modules and libraries.

• Debugging and Profiling Tools:

Additional tools that help developers identify and fix errors, as well as analyze the performance of the compiled code.