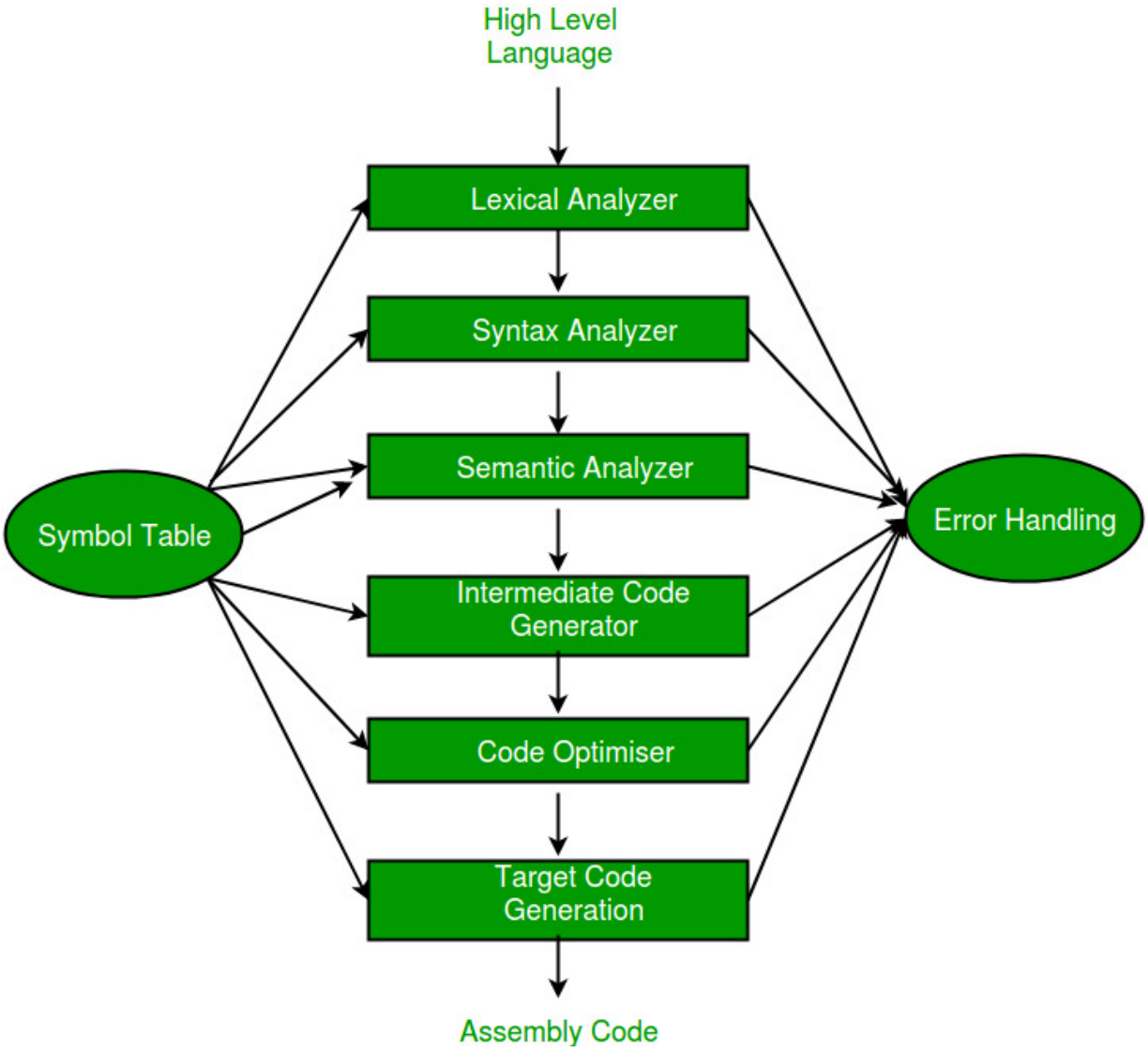
LEXICAL ANALYZER



Abu Shahid **Atharva Pandey**

B20S003 **B20CS007**

INTRODUCTION

We designed an elemental analyzer in C++ to accept our language. The analyzer removes whitespaces from the input code, ignores text within comments, and recognizes tokens following the lexeme description given in the assignment problem.

EXECUTION

ZIP contents:

1. Block_diag.png
2. def.h
3. driver.cpp
4. exec.sh
5. input.in
6. pa_1.out
7. PA1.pdf
8. symbol_table_1.out

To execute the program, simply execute the `exec.sh` executable.

./exec/sh

It internally compiles the driver.cpp and creates FSM executable. It accepts the input.in using redirection to write input.in in stdout and updates/initializes pa_1.out and symbol_table_1.out

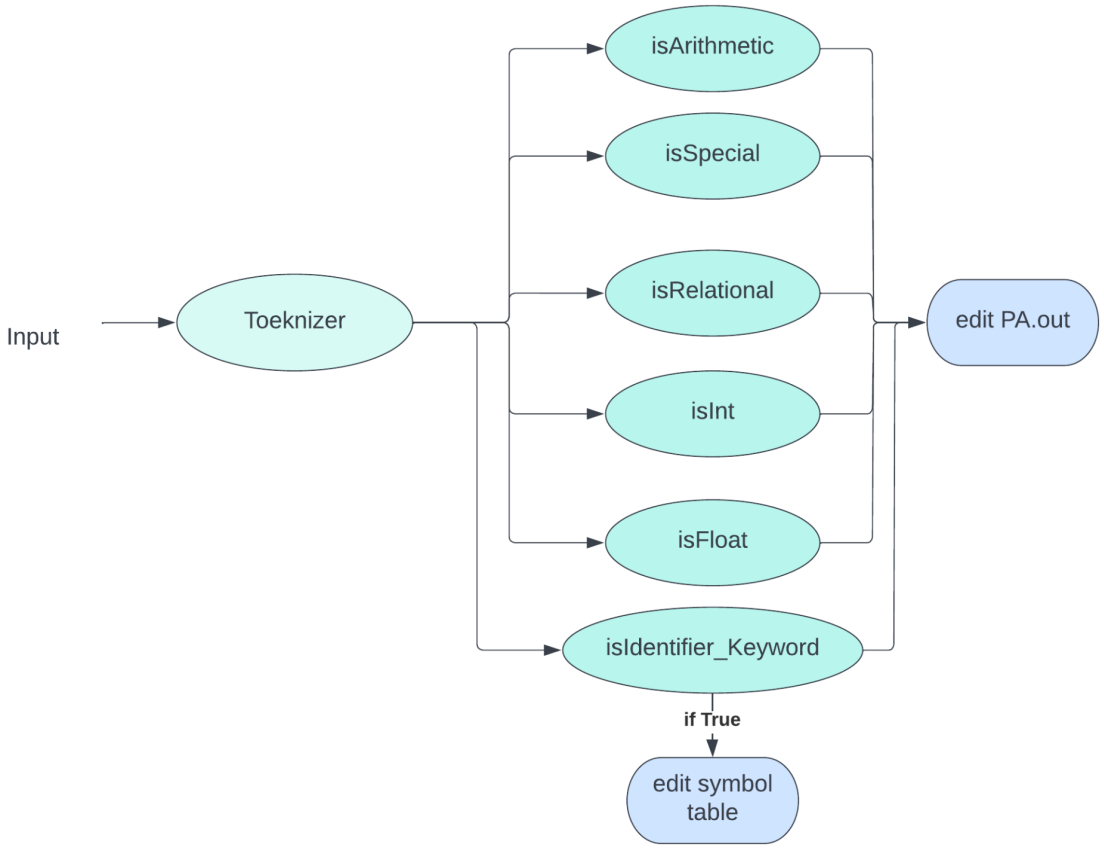
Manually, one can run the following commands:

g++ driver.cpp -o FSM

./FSM < input.in

IMPLEMENTATION

- Our driver executable takes input through redirection and tokenizes it with various delimiters, returning a vector of tokens. Removal of white spaces and comments is taken care of; during this initial tokenization step.
- Second, we iterate through each token in our token list and pass it to the following different FSMs we have made to identify each token:
    - isSpecial(), isRelational(), isArithmetic(), isInt(), isFloat(), isIdentifier_Keyword()
- Furthermore, after identifying if a given token is Identifier or Keyword; isIdentifier() and isKeyword() are separately called to make necessary amends and appends in the symbol table.
    - If it is a keyword, we push back the token and 0 identifier, else a 1.



- Individual FSMs for each token can be better understood via code file **driver.cpp.**