Marathi-English Mini Compiler Documentation

Project Title:

Marathi-English Mini Compiler with 3-Address Code Generation

Objective:

To design and implement a simple compiler that:
- Supports a mix of English and Marathi keywords.
- Parses arithmetic and assignment operations.
- Converts code into intermediate Three Address Code (TAC).
- Demonstrates how compilers tokenize, parse, and translate code.

Technologies Used:

Flex - Lexical analyzer (lexer)
Bison - Syntax analyzer (parser)
C - Integration and logic (TAC)
Terminal (macOS/Linux) - Compilation

Language Features Supported:

- पूर्णांक : Variable declaration (int)
- dprint : Print statement
- = : Assignment
- + : Arithmetic addition
- Semicolon ; : End of statement
- Identifiers : User-defined variables
- Literals : Numeric constants (integers)

Example Code (Marathi-English Hybrid):

```
पूर्णांक x;
x = 10 + 20;
dprint x;
```

Compiler Architecture:

1. Lexical Analysis (Flex)

- Tokenizes keywords, identifiers, numbers, and symbols.
- Converts Marathi keyword 'पूर्णांक' to token INT
- Converts 'dprint' to token DPRINT

2. Syntax Analysis (Bison)

- Defines grammar rules for:
  - Declaration
  - Assignment
  - Arithmetic expressions
  - Print statements

3. Semantic Action (C code)

- Generates three-address code using temporary variables (t0, t1, etc.).
- Outputs actions like 't0 = 10', 'x = t0', 'dprint x'.

Sample Output (3-Address Code):

Input:
पूर्णांक a;
a = 5 + 3;
dprint a;

Output:
Declare a
t0 = 5
t1 = 3
t2 = t0 + t1
a = t2
dprint a

How to Compile:

1. Create source files:
   - parser.y – grammar rules
   - lexer.l – token patterns
   - tac.c – TAC generation logic
   - main.c – main function

2. Generate parser & scanner:
   bison -d parser.y
   flex lexer.l

gcc -o compiler main.c lex.yy.c parser.tab.c tac.c

3. Run the compiler:
./compiler

File Structure:

compiler/
├── main.c
├── parser.y
├── lexer.l
├── tac.c
├── parser.tab.c / h (generated)
├── lex.yy.c (generated)
└── compiler (output executable)

Example Test Cases:

Test 1:
पूर्णांक a;
a = 5 + 2;
dprint a;

Test 2:
पूर्णांक a;
पूर्णांक b;
पूर्णांक result;
a = 2 + 3;
b = a + 4;
result = a + b;
dprint result;

Limitations:

- Only supports + operator (no -, *, /).
- No control structures (if, while, etc.).
- No type checking or error recovery.

Future Enhancements:

- Add support for:
  - Other arithmetic and logical operators.

- Control flow (if, while, for).
- Strings, floats, and arrays.
- Add a symbol table for scope management.
- Implement a full interpreter backend.

Output of compiler :