Compiler Design Project

Semester: Fall 2023

Project Title: Mini Compiler

Project Description: In this project, you will develop a mini compiler that performs lexical analysis, syntax analysis, and basic semantic analysis for assembly (MIPS) programming language.

Project Phases:

1. Lexical Analysis
2. Syntax Analysis
3. Semantic Analysis

Notes:

• You should identify the tokens classes in a way that will help you in the following phase (parser)

• You can use any programming language you prefer for building the scanner.

• The project deadline for phase 1 is 2/12 and discussion during (3/12 – 7/12)

• Each team should be 3 to 4 members

• A discussion will be with all group members; all members should participate in implementation.

• **Very important: any plagiarism detected will lead to losing the project marks**

The details of each phase are given below.

Lexical Analysis:

- Implement a lexical analyzer (scanner) that reads the source code character by character.
- Define a set of regular expressions to represent different token types (e.g., identifiers, registers, operands, instructions, etc.).
- Develop a token recognition mechanism that maps input patterns to token types.
- Build a symbol table to store information about identifiers encountered during tokenization.

Input (Source Code)

Example:

```
    lw $t0, num1        # Load num1 into register $t0

    lw $t1, num2        # Load num2 into register $t1

    add $t2, $t0, $t1   # Add num1 and num2, store result in $t2

    sw $t2, sum         # Store the sum in memory location 'sum'
```

Output (Tokens and Symbol Table)

Tokens:

Token: Load, Lexeme: lw

Token: Register, Lexeme: $t0

Token: Comma, Lexeme: ,

Token: Identifier, Lexeme: num1

Token: Load, Lexeme: lw

Token: Register, Lexeme: $t1

Token: Comma, Lexeme: ,

Token: Identifier, Lexeme: num2

Token: Add, Lexeme: add

Token: Register, Lexeme: $t2

Token: Comma, Lexeme: ,

Token: Register, Lexeme: $t0

Token: Comma, Lexeme: ,

Token: Register, Lexeme: $t1

Token: Store, Lexeme: sw

Token: Register, Lexeme: $t2

Token: Comma, Lexeme: ,

Token: Identifier, Lexeme: sum

Symbol Table:

- Name: num1, Type: memory address
- Name: num2, Type: memory address
- Name: sum, Type: memory address