CSL302: Compiler Design

Lab-7 (2025-26-M Semester)

Due Date: October 11, 2025

Instructions

- Prepare a zip file and upload your solutions on canvas
- Include a readme file

Question

In the lab exam-1, you were asked to write the lexical analyzer following assembly code. In this lab, you are required to implement the syntax analyzer for the same language. The features are listed below for ease of reference. You can reuse the lexical analyzer code that was submitted.

1. **Opcodes (instructions)** - Recognize a fixed set of mnemonics: `MOV, ADD, SUB, MUL, DIV, LOAD, STORE, JMP, CMP, HALT`.
2. **Labels** - User-defined names containing one or more captial alphabets (e.g., `LOOP`).
3. **Registers** - Recognize register names: `R0, R1, R2, ..., R9`.
4. **Immediate Numbers** - Integer values beginning with # (e.g., `#10`, `#255`, `#-5`).
5. **Delimiters / Special Symbols** - `,`, :
6. **Comments** - Everything after `;` on a line should be ignored (including ;).
7. **Errors** - Anything other than above tokens should be treated as error.

Rules:

- An assembly program contains zero or more statements.
- Each statement can be one of the following: Arithmetic statement, Conditional statement, memory statement, and Halt statement
- The arithmetic statement can starts with MOV, ADD, SUB, MUL, DIV, followed by 2 operands, which are separated by commas. The first operands should be REG, the second and third operand can be REG or CONST. Examples are MOV R1, #10, ADD R2, R1
- Conditional statements start with JUMP followed by LABEL
- Memory statement can start with LOAD or STORE followed by two operands. The first operand is a Register second operand is a memory location specified using Immediate number.

- Halt statement starts with HLT_OP and does not have any operands. Each statement can be optionally prefixed with LABEL:

Example Input

```
START: MOV R1, #10      ; load 10 into R1
ADD R2, R1        ; R2 = R2 + R1
SUB R3, #5
JMP START
HALT
```

Expected Output

```
The above program is syntactically correct.
```