Lab 07 LC-3 Assembler

1 Overview

This lab involves implementing a basic assembler for the LC-3 assembly language. Before you start, ensure you have read Section 7.3 (p. 240) of the textbook. Your task is to create a toy assembler, which will be tested using specific `.asm` files (not disclosed to you).

The purpose of this lab is to deepen your understanding of the assembly process. Therefore, you can disregard certain complexities typically associated with `.asm` files.

2 Requirements for the Assembler

Your assembler should be able to process `.asm` files that adhere to the following constraints:

1. **Legal Files**: The files will not contain illegal opcodes or address conflicts.
2. **Uppercase Instructions**: All English characters in the instructions are uppercase (e.g., `ADD`, `.ORIG`).
3. **Number Prefixes**: Intermediate numbers in the file will have either a `#` or `x` prefix.
4. **No Comments**: The `.asm` files will not contain code comments.
5. **Limited Pseudo Operations**: Only `.ORIG`, `.END`, `.FILL`, `.BLKW` and `.STRINGZ` will be used.
6. **No Instruction Aliases**: Use the standard opcodes (e.g., `TRAP x25` for `HALT`).
7. **Single `.ORIG`**: Only one `.ORIG` instruction per file.
8. **Formatting**: No unnecessary spaces or tabs. Commas must be followed by a space.

   For example:

   ```
   ADD R1, R1, R2
   ```

   These formats will not appear:

   ```
       ADD R1, R1, R2

       ADD  R1,  R1,  R2

   ADD  R1, R1,R2
   ```

9. **Label Placement**: Labels appear directly before an instruction, not on separate lines or with added colons.

   For example, an instruction with label `MAIN` will be:

   ```
   MAIN ADD R1, R1, R2
   ```

These formats will not appear:

```
MAIN: ADD R1, R1, R2
MAIN:
ADD R1, R1, R2
MAIN
    ADD R1, R1, R2
```

A sample testcase is provided with this document.

3 Translation Requirements

- Follow the translation guidelines as per Figure A.2 in the textbook (p. 656).
- The first line of your code should reflect the start address of your program.

  For example, if your program is:

  ```
  .ORIG x3000
  ...
  ```

  The result of your translation file should be:

  ```
  0011000000000000
  ...
  ```

4 Submission Format

Develop the assembler in C or C++ (C++20 standard). If you want to, use standard libraries only. Avoid external libraries like boost.

Include in your submission a `.c` or `.cpp` file, along with a PDF report detailing your approach and ideas.

Submit your work in a ZIP file named `Lab7_PB22XXXXXX.zip`, containing:

- assembler.c/cpp
- report.pdf

5 Compilation and Testing

- We will compile your program using

  `gcc assembler.c -o assembler` or

  `g++ assembler.cpp -o assembler -std=c++20`
- Your program should accept command arguments and be capable of processing an input file and outputting to a specified file.

  For example:

  `./assembler test_in.asm test_out.txt`

  Then your program should use `test_in.asm` as input file and output results to file `test_out.txt`