Lab07 Report

崔士强 PB22151743

January 12, 2024

This lab is completed in MacOS.

1 Purpose

The purpose of the program is to implement a simple LC-3 assembler in C++.

Anticipated outcomes: A `.txt` file including the machine language code translated from the input `.asm` file

2 Principles

2.1 Two-pass process

The assembling process is a two-pass process. The first pass reads the labels and creates a symbol table. The second pass reads the code and translate it into the machine language.

In order to identify the labels, a vector including all the operations is generated: `vector<string> instSet`. If a line is not started with an operation, store the first word as a label in the map `map<string, int> symbol_table`.

In the second pass, the code is translated into machine language line by line:

```
PC = 0;
for(const auto &line : lines){
    if(line == ".END") continue;
    PC++;
    output_lines.push_back(translate_instruction(line, symbol_table, instSet, PC));
          // Translate the instruction in the second pass
}
```

The `PC` here is not the value of program counter. It started with 1 in order to simplify the process.

2.2 Translation

After identifying the operation, the translation process can be broken up into three parts:

1. Registers

2. Immediate values
3. Labels

For each of them, a function is defined:

```
void FetchRegister(string &instruction, string &machine_code);
void FetchImmediate(string &instruction, string &machine_code, const int len);
void FetchLabel(string &instruction, string &machine_code, const map<string, int>
    symbol_table, const int PC, const int offset_length);
```

For registers and immediate values, the work is mainly converting strings of decimal or hexadecimal numbers to fixed-length strings of binary numbers. And for labels it's mainly computing offsets.

3 Procedure

3.1 Bugs encountered

During execution, I found that negative numbers could not be converted to 2's complement binary representation correctly. The reason was that the '−' before a number was not deleted. So sign-extending this number would give a result like `------110`.

Solution: add the following code:

```
if(s[0] == '-'){
      negative = true;
      s.erase(0, 1);
  }
```

3.2 Chanllenges

Given that:

1. I've never learned C++ before.
2. Implementation using C is more complex due to the absence of useful libraries like STL.

A major challenge in this lab is getting familiar with C++. If given more time, maybe I'll try using regex, which could simplify the work.

4 Results

Results are shown below:

Listing 1: test_in.asm

```
.ORIG x3000
MAIN LD R1, DATA
TEST ADD R1, R1, #-10
```

```
4   BRZP TEST
5   ST R1, MEM
6   TRAP x25
7   DATA .FILL x1234
8   MEM .BLKW #2
9   .STRINGZ "HelloWorld1337"
10  .END
```

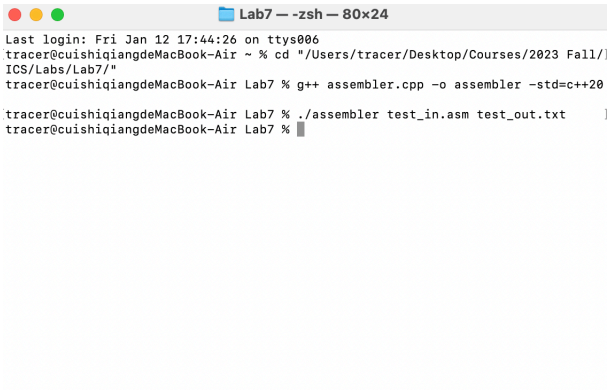

Figure 1: Command

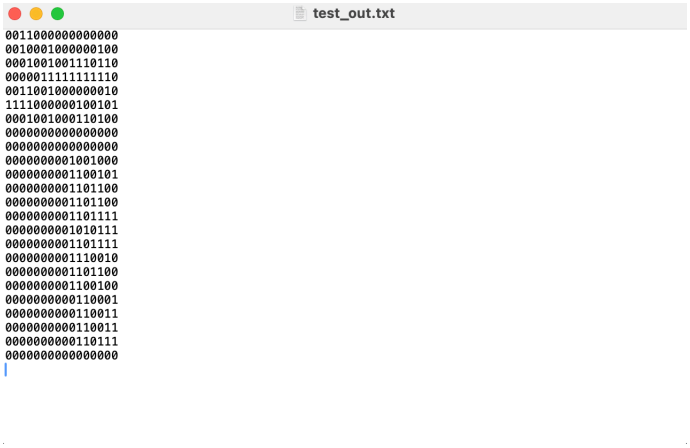

Figure 2: Result

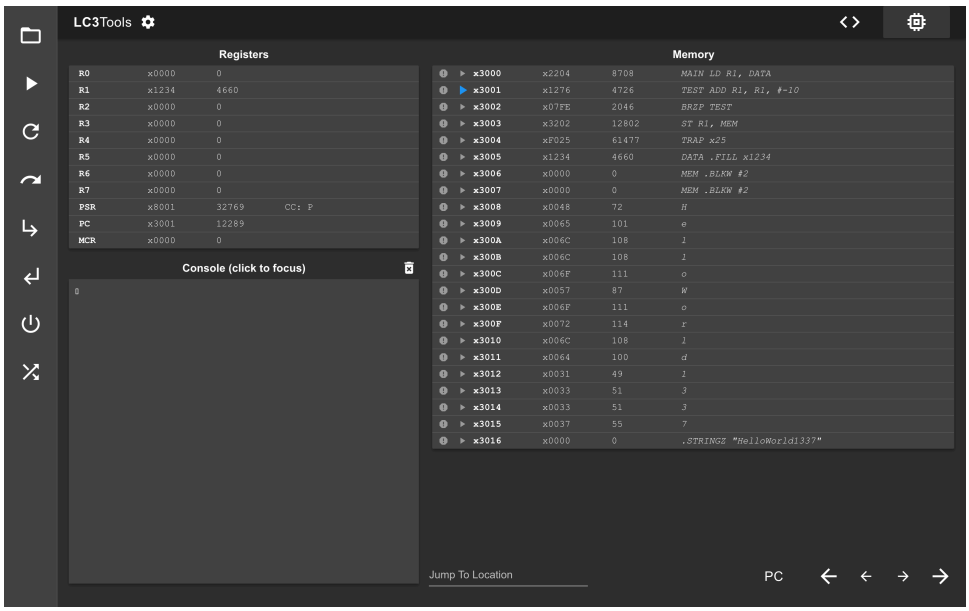

Figure 3: Assembly result in LC-3 Tools

The machine code in output file is correct.