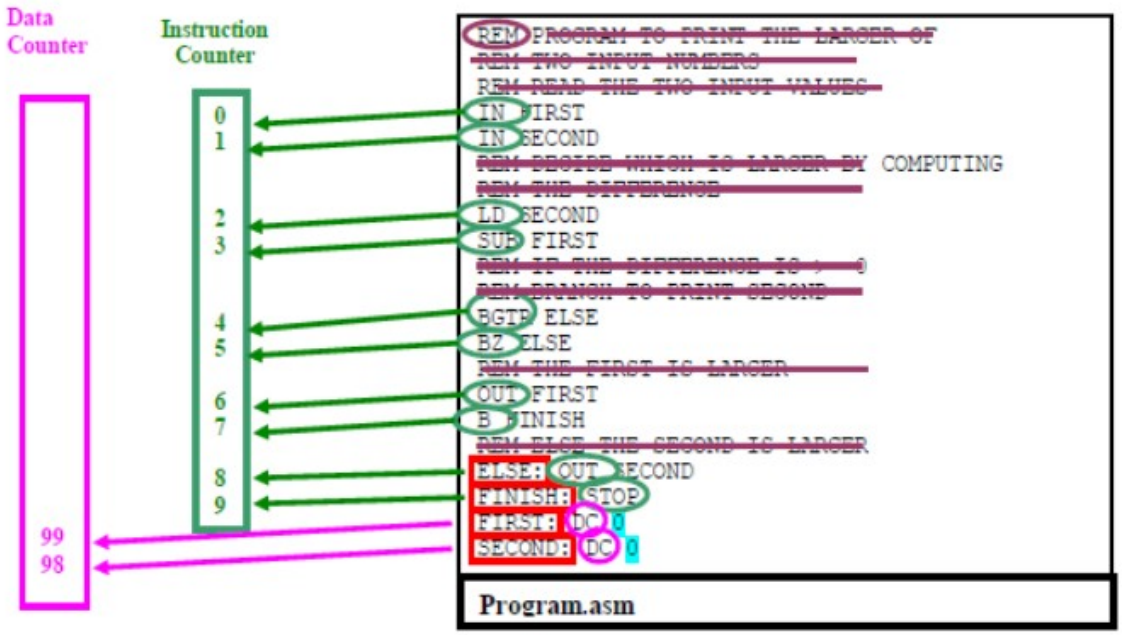
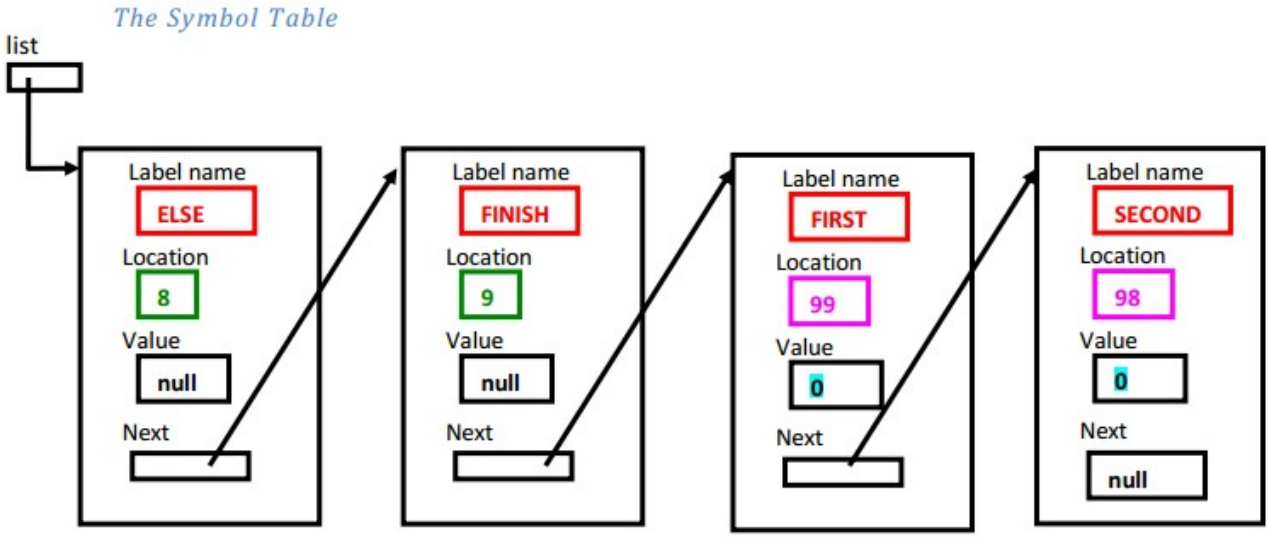
Lab 7: The First Pass

Building the Symbol Table





Step 1: Create the classes to store the symbols

Building the symbol table requires you to have a list structure, as we cannot assume we know how many labels (variables or instructions) will be used in a program. This implies that we must have:

1. A class Symbol which stores the information a symbol must keep i.e. the label name, the memory location associated with the label, and the value (if it is a variable). Note that value can be defined as an object of type Integer so that it can be fixed to null. This class must provide all accessors/mutators as well as the toString method.

2. A class SymbolTable which provides the means to create the list, and add elements to the list. In order to check it is working, it is also worthwhile to have the toString() method to print the content of the list. This class should implement the SymbolCollection interface.

Create a small test program to check that you can create a list, and add some symbols to it correctly.

Step 2: Create a compile() method in the Computer Class

The TestComputer class provides the header for the compile method:

```
cpu.compile("program.asm");
```

The method compile must take a String (the file name) as a parameter and must build the symbol table before doing a second pass to create the executable file.

Algorithm:

It is easiest to break down the compilation process into two major steps:

1. First pass
   - Build symbol table using ASM file.
   - Return the symbol list.

2. Second pass
   - Use symbol list and ASM file to .
   - Write instructions in executable file.

In your computer class, create 2 methods named firstPass and secondPass. You will also need to create multiple “helper” methods to aid in simplifying your code. These helper methods are defined in the SymbolCollection interface. Based on the compile algorithm, we can deduce that firstPass returns a SymbolTable object and needs a file name as a parameter, so that compile consists of:

```
SymbolTable symTbl = firstPass(filename);
secondPass(symTbl, filename);
```

To test that the SymbolTable is correctly built from the file, we would need to print the symTbl object:

```
System.out.prinltn(symTbl); //this line is just for testing
```

Algorithm for the firstPass() method

The The algorithm for the firstPass method is based on what has been described in the lab for building the symbol table:

1. Open ASM text file.

2. Read a line from the ASM file.
3. Extract the first word in the line.
4. Is the first word a comment or a blank line?
    - Yes
      o Skip this line and go to step 2.
    - No
      o Continue with algorithm.
5. Is the first word a label?
    - Yes
      o Extract second word.
      o Is the second word a variable?
        - Yes
            - Save in symbol table. The label name is the first word, the location is the current value of the data counter, and the value is zero.
            - Decrement the data counter.
        - No
            - We have found a regular label. Save in symbol table. This label name is the first word, the location is the current value of the instruction counter, and the value is null.
            - Increment the instruction counter
    - No
        - Increment the instruction counter.
6. Have we read the last line in the ASM file?
    - **Yes:**
      o Close the ASM file.
      o Return the populated symbol table.
    - **No**
      o Repeat from step 2.

Notes:
The ASM file is a text file, so the Scanner class or the FileReader class can be used to read it.
Use of the debugger is highly recommended.