```
package nand2tetris;

import java.io.*;
import java.util.*;

public class HackAssembler {

    private BufferedReader r;
    private PrintWriter w;
    private SymbolTable symbolTable;
    private Parser parse;
    private Code translate;
    private Vector<String> program;
    private int programLength;
    private String line; // Current line.

    private static final String assemblyFile = "/home/magiwanders/Documents/Studio/
SelfStudy/ComputerArchitecture/nand2tetris/projects/06/pong/PongL.asm";

    public HackAssembler() {
        symbolTable = new SymbolTable();
        parse = new Parser();
        translate = new Code();
        program = new Vector<String>();
        initializeIO(); // Creates reader and writer to .asm and .hack files
respectively.
        execute();
    }

    private void initializeIO() {
        try {
            r = new BufferedReader(new FileReader(new File(assemblyFile)));
            w = new PrintWriter(new FileWriter(new
File(assemblyFile.replaceAll(".asm", ".hack"))));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void execute() {
        loadFile(); // Also removes spaces and comments.
        scanLabels();
        finalScan();
        exit();
    }

    private void loadFile() {
        try {
            while(true) {
                line = r.readLine();
                if (line == null) break; // File ended.
                clean(); // Removes spaces and comments from each line.
                if(!line.isEmpty() && line != null) {
                    program.add(line);
                }
            }

            programLength = program.size();

            printProgram("After loadFile()");

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```



```
    private void clean() {
        if(line.equals(null)) return; // Ya never know.
        line = line.replaceAll(" ", ""); // Removes spaces
        line = removeComments(line); // Removes comments
    }

    private String removeComments(String line) {
        int index = line.indexOf("//");

        if(index != -1) { // In case there is a commment.
            line = line.substring(0, index);
        }

        return line;
    }

    private void scanLabels() {
        String label;
        int index1, index2;

        for(int i=0; i<programLength; i++) {
            line = program.elementAt(i);

            index1 = line.indexOf("(");
            index2 = line.indexOf(")");

            if(index1 != -1 && index2 != -1) { // Se c'è una parentesi
                label = line.substring(index1+1, index2);
                symbolTable.add(label, i); // Gli assegno la linea della label
perchè toglierò quest'ultima e l'istruzione successiva prenderà il suo posto.
                System.out.println("i="+i+" Trovata label: "+label+". Memorizzata
in: " + (i));
                program.removeElementAt(i);
                i--; // Torna indietro perchè ho tolto la linea corrispondente
alla label.
                programLength = program.size();
            }
        }

        printProgram("After scanLabels()");
    }

    private void finalScan() {
        for(int i=0; i<programLength; i++) {
            line = program.elementAt(i);
            if(line.indexOf("(")!=0) {
                if(line.indexOf("@") == 0) handleAInstruction();
                else handleCInstruction();
            }
        }

        printProgram("After finalScan()");
    }

    private void handleAInstruction() { // @value
        String valueString = parse.AInstructionInt(line);
        int value;

        try {
            value = Integer.parseInt(valueString);
        } catch (NumberFormatException e) { // It is a variable.
            value = symbolTable.retrieveValue(valueString);
        }

        String binaryValue = Integer.toBinaryString(value);
        for(int i=0; binaryValue.length()<16; i++) {binaryValue = "0" +
```

```
binaryValue;}
        w.println(binaryValue);
    }

    private void handleCInstruction() { // dest = comp ; jump
        String dest = parse.dest(line);
        String comp = parse.comp(line);
        String jump = parse.jump(line);
        System.out.println("Trovata istruzione | " +dest+"="+comp+";"+jump);
        String accccc = translate.comp(comp);
        String ddd = translate.dest(dest);
        String jjj = translate.jump(jump);

        w.println("111" + accccc + ddd + jjj);
    }

    public void exit() {
        try{
            w.close();
            r.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String [] args) {
        HackAssembler hackAssembler = new HackAssembler();
    }

    private void printProgram(String caller) {
        System.out.println( "[" + caller + "] Corrente stato interno del
programma: ");
        for(int i=0; i<programLength; i++) {
            System.out.println(program.elementAt(i));
        }
    }


}
```