CSCI-410 Python Assignment - PY-01

Over the course of the next few weeks you will write a "disassembler" for the Hack computing platform. The primary goal of these assignments is not to obtain a disassembler, though it is certainly believed you will benefit from developing one, but rather to at least be introduced to several aspects of the Python scripting language that will be needed and/or useful in developing the software that is part of the Nand2Tetris projects starting at about the middle of the semester.

These topics include

- File I/O
- Working with strings
- Implementing finite state machines
- Basic XML
- Using dictionaries
- Writing classes
- Processing command line arguments
- Creating and importing class libraries
- Working with directories
- Regular expressions

In this first assignment your goal is very simple. You will open an ASCII text file named machine.hack, which contains executable code for the hack processor. Normally, these files would be stored in a binary format, but the ECS Nand2Tetris project uses all ASCII based text files to facilitate the educational objectives.

The following is the contents of one of the hack files supplied with ECS Project #06.

```
0000000000000010
1110110000010000
0000000000000011
1110000010010000
0000000000000000
1110001100001000
```

Each line in a hack file consists of a single instruction. Each instruction is a 16-bit word represented in binary using the characters '0' and '1' as symbols. There are no comments or blank lines allowed. There may be whitespace after a line before the line return.

Eventually, your disassembler will produce an assembly language source code file that can be assembled and run on the hack platform. For now, our goal is simply to place the executable code into comments in an assembly source code file after doing some very minor processing.

Your output file is to be named assembly.asm and will have the following format, using the hack code above:

CSCI-410 Python Assignment - PY-01

```
// 00000: 0000 0000 0000 0010
// 00001: 1 11 0 110000 010 000
// 00002: 0000 0000 0000 0011
// 00003: 1 11 0 000010 010 000
// 00004: 0000 0000 0000 0000
// 00005: 1 11 0 001100 001 000
```

Each line begins with the end-of-line comment delimiter, which consists of two forward slashes.

This is followed, after a space, by the 5-digit, zero-padded decimal address of that the instruction on that line. Instruction addresses begin with 0 and increase in order for each instruction in the file. The address is immediately followed by a colon and a space.

At this point the format varies depending whether the instruction is an A-type instruction or a C-type instruction. Any instruction whose first bit is '0' is an A-type instruction while all others are C-type instructions.

The A-type instructions will be parsed into four, 4-bit groups, each separated by one space.

The C-type instructions will be parsed into six groups, each separated by a space, consisting of lengths {1,2,1,6,3,3}, in that order.

There shall be no whitespace after the end of each line. Furthermore, the last line in the file will be an instruction, meaning that there should be NO newline character at the end of this line.

Submission

In a manner similar to the ECS project, place all files needed for this assignment into a directory named PY01 and zip up the entire directory into a zip file of the form:

CS410_UserID_PY_01.zip

GRADING RUBRIC – 20 pts

5 pts	**Effort**
3 pts	**Lines formatted as valid EOL comments.**
2 pts	**Addresses correct and correctly formatted**
4 pts	**A-type instructions parsed correctly**
4 pts	**C-type instructions parsed correctly**
1 pts	**No whitespace at end of lines**
1 pts	**No newline after last instruction**
-1 pts	**Incorrect header comments in Python script.**
-1 pts	**Incorrect submission (filename, etc).**