log_book

2015/06/07
Shunsuke Ono a1675327
This is a log_book for assignment4.

Initial thoughts od solutions.

- assembler is what translates assembly code to binary codes.
- For the translation, there are two parts:
  - 1. Translate assembly command with the straightforward relationship, which is not complexed.
  - 2. Dealing with user defined symbols of labels and variables, which is comparetively difficult. I can solve this with symbol table.

3 types of symbol

- predefined symbols
- label symbols
  - which indicates a ROM location of each instruction
- variable symbols
  - which indicates a RAM location of each variable

=> So I need to set predefined symbols to symbol table when I initialize it. For label, because you want to use label even at a command which is located before the label declearation, you need to pass through the codes once and register all labels into symbol table.

2 passes

1st pass

- march each line and dont generate any code but set symbols to symbol table

- you will pick up only labels but variables
- you need to keep counting the command number, which will be used for allocating each code to memory location. then you can specify ROM address that each label should have.
- when encountering label for the first time, add an entry (Xxx, n) where Xxx means the symbol and n memory location.
- variable will be dealed with in 2nd pass

2nd pass

- when encountering symbol, check if your symbol table contains it already or not.
  - if yes, retrive the address of variable and use it for translation process
  - if no, add an entry for the variable (it should be variable because you alrady registered all labels in 1st pass so that you should have each label in your table).
- parse and code each code with actual address, then translate it.

Plans for inplementations and Procedure

4 modules

- Parse module: parse input
- code module: generalte binary code corresponding to each assembly command
- symbol table: manage symbol
- main program: manage whole process

then implement an assembler in two steps:
- 1. write an assembler which can translate assembly code without any symbols
- 2. enable it deal with symbols with symbol table

with symbols

First, I need to implement symbole_table module to deal with user defined table.

SymbolTable

- constructor
- addEntry: add an entry(Xxx, n) to symbol table
- contains: check if the table has specified symbol
- getAddress: retrive an address of specified symbol

Mistakes I made and found

- Resolve some typing misstakes when debugging.
- I misunderstood the variable address starts from 1024, but it depends on each platform and starts from 16 for Hack.
- I forgot to add predefined variables to symbol table.