Objective.

Write an Assembler program, named myassembler

<conventions>

- .hack → .asm
- constants and symbols ⇒ lookup symbol table and get meaning or define
- Comments → with // at the beginning of sentence
- white space → ignored
- code conventions

<implementation>

The Parser Module

Constructor input file
hasMoreCommands
advance
commandType
symbol
dest.
comp
jump

The Code Module

dest
comp
jump

⇒ create class for each module
and method for each routine

Scanned by CamScanner

parser:: ~~set_instructions()~~
~~is_instruction()~~

//

First, make array of string ∈ instructions.

Skip all blank lines and comments
↙ 1 case ↙↙ 2 cases

line.length() == 0 // ~~ @Xxx // ≈

Parse

First get only instructions by removing white space and comments.

Get array of instructions

So I made some methods, set_instructions remove_comment is_instruction to get an array of instructions.

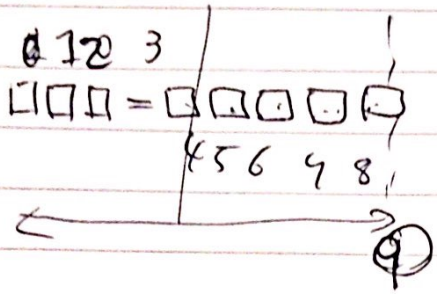
1. the program opens an output file named ㉠.hack

2. the program marches through the lines in supplied ㉡.asm file.
For each C-instruction, create 16-bit word

3. write word on the .hack file.
For each A-instruction, resolve decimal constant to binary number and write the 16-bit word into .hack file

.asm ⇒ .hack

Scanned by CamScanner



Code

For each section in construction, return binary code by Code module.

- dest
- comp
- jump

Assembler

For using the combination of parser module and code module, you need to implement main module, which I named assembler.cpp

Scanned by CamScanner