Assembler

Tom Kelliher, CS 220

Chapter 6, Textbook

- Section 6.1: What is a symbol?

  What is symbol resolution/binding?

  What is the function of the assembler?

- Section 6.2: State a simple, algorithmic rule for distinguishing if a given string is a constant or a user-defined symbol.

Chapter 6, Slides

- Slide 3: Are all Hack assembly instructions (A- and C-instructions and labels) assembled into machine instructions? Explain.

- Slide 4: Symbols can't begin with a digit, simplifying the recognition of a symbol and a variable in an A-instruction.

- Slide 5: There are four possible combinations of fields in a C-instruction:

  - `comp`
  - `dest=comp`
  - `comp;jump`
  - `dest=comp;jump`

  The presence/absence of `=` and `;` denote the presence/absence of the `dest` and `jump` fields, respectively. The `comp` field is always present.

- Slide 6: Name and describe the five types of symbols in Hack assembly.

  Note that `@END` in the third line of example Hack assembly represents a forward reference to the `(END)` label near the end of the example. This is why the assembler requires two passes.

- Slide 8: Given some Hack assembly code, produce the corresponding symbol table.

- Slide 9: Give a specific reason, referencing symbol table functionality, for why a single pass Hack assembler is impossible to write.

- Slide 10: Describe what is done in each of the two passes of the assembly process. Be specific.

- Slide 11: Given some Hack assembly code, completely assemble it into machine code.
- Slide 12: See the textbook for a detailed description of the proposed implementation.