**Computing Labels**

Labels are symbolic names for targets of branch instructions

Two-Pass Approach

* generate sequence of instructions with unknown destinations
* once lengths of code are known, resolve all labels
* emit the code

Two main kinds of jumps:

* absolute: indicate address from some fixed position
* relative: indicate address as an increment over current program counter
  + positive increment: jumping forward
  + negative increment: jumping backward

**Assemblers**

A minimal infrastructure for writing machine code

* allow ASCII names for instructions instead of operation codes
* avoid having to know labels for jumps (what we needed now)
* store constants for strings and other large data (give back the address of that data)

Examples:

* Our [cafebabe](http://lara.epfl.ch/w/compilation:cafebabe) library enables Scala to be used as an assembler for Java bytecodes
* [GNU Assembler](http://en.wikipedia.org/wiki/GNU%20Assembler)
  + take small C file test.c, do ‘gcc test.c -S’
  + run the assembler by ‘as test.s’