

# Homework # 1

## due 1/29/08

(This homework, as in all homeworks, must be completed by each person separately. You may discuss the problems with each other and help each other, as long as you write the names of the people you worked with prominently on your homework. You may not use xerographic or electronic copying; please do not send help in the form of emailed or xeroxed answers.)

### 1 GCD

The textbook gives two MIPS assembly versions for a GCD program and shows a more readable version in Pascal. Write a feature `gcd` and call it from the `Main` constructor with some sample values. Attach the Cool program and the compilation of the GCD feature to your homework. Just print out the assembly language for the GCD feature, **not** the entire compilation. Part of this assignment is figuring out where this code is.

The Cool compiler generates very inefficient code for arithmetic operations. To subtract two numbers, `a-b`, it first loads one operand, then stores it again. Then it loads the other operand, copies it and then loads the first operand. Then it loads the raw integer from each integer object, subtracts it and stores it in the copied integer. Outline the 6–8 instructions where this occurs in the generated assembly code.

### 2 Errors

Solve Exercise 1 (p. 26–27) using Cool: give examples of a lexical error, a syntax error (“parse error”), a static semantic error, and a dynamic semantic error (one not caught by the compiler). Give the actual output that results in each case.

A lexical error results in a message concerning the pseudo-token `ERROR`; it is not a “parse error.” Misspelling a keyword is not reported as a lexical error; make sure your example of a lexical error leads to a message including the `ERROR` pseudo-token.

A dynamic semantic error is where the program compiles but when it runs, it stops before the main program finishes. Usually an error message is printed.

### 3 Compiler Construction

Suppose you were interested in the Lisp programming language. You found out that you can get a good lisp compiler in open source, `OpenCL`, but unfortunately `OpenCL` is written in Lisp and it generates portable C. You also found a Lisp interpreter `IL++` written in C++. Assume you have a working C++ compiler for your computer. How can you use the pieces that you have to get a working stand-alone Lisp compiler (that doesn’t require an interpreter to run)?