EE 3755 **Homework 4** **Due: 12 November 2001**

See `http://www.ece.lsu.edu/ee3755/ref.html` *for documentation on MIPS and the SPIM MIPS simulator. Alternate instructions can be found in Appendix A of the Patterson & Hennessy text. The links are clickable when this assignment is viewed with Acrobat Reader.*

Copy files `/home/classes/ee3755/com/s/hw04.s` *(the solution template) and* `/home/classes/ee3755/com/s/hw04p0.s` *into a subdirectory named* `hw` *in your class account. File* `hw04p0.s` *contains instructions for running the SPIM simulator. When these instructions are followed the SPIM simulator is run with the* `-notrap` *and* `-delayed_branches` *switches, and a version of the simulator implementing the* `clz` *instruction is used. If you choose to use another installation of the simulator be sure to start it using those switches and replace the* `clz` *instructions in the solution template with a call to the completed* `countlz` *routine (see Problem 1).*

Problem 0: This solution to this problem will not be collected graded, **but do it anyway!** File `hw04p0.s` contains a buggy MIPS procedure, `pop`, that's supposed to find the population of an integer. (An integer's population is the number of 1's in its binary representation.) The file also contains a startup routine that runs `pop` on three different integers, moving the incorrectly computed population into registers `s0`, `s1`, and `s2`.

(*a*) Following the instructions in the file, run the code and verify that it doesn't work.

(*b*) Debug the `pop` count module.

Problem 1: Write a MIPS assembly language procedure that counts the number of leading zeros of the value in `a0` and puts that count into `v0`. (Remember that MIPS registers are 32 bits.) For example, if 1 (written another way $00000000000000000000000000000001_2$) is in `a0` then 31 should be put in `v0`. If $ffff_{16}$ is in `a0` then 16 should be put in `v0`. Put the code in the appropriate place in the solution template after the line labeled `countlz` in hw04.s. The template is set up with a routine to test `countlz`. Do not use the `clz` instruction.

Problem 2: Write a MIPS assembly language procedure that converts a 32-bit signed integer in register `a0` into an IEEE 754 single-precision floating point number and puts it in `v0`. The procedure must round the integer towards zero (for most solutions, this is equivalent to not doing any rounding at all). Put the code in the appropriate place in the solution template: after the line with label `itos` in `hw04.s`. The template is set up with code to test `pops`. Do not use MIPS' floating-point instructions.