# CSCI 2321 (Principles of Computer Design), Spring 2015

# Homework 5

**Credit:** 20 points.

## 1 Reading

Be sure you have read all assigned sections of Chapter 2 and Appendix A.

## 2 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 2321 homework 5"). You can develop your programs on any system that provides the needed functionality, but I will test them using the SPIM simulator on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in. (Specifically, I will test from the command line with a command of the form `spim -f yourcode.s`.)

1. (10 points) Write a complete MIPS assembler program for the SPIM simulator that prompts for a sequence of integers ending with 0 and prints their sum. It should behave like this C program:

```
#include <stdio.h>

int main(void) {
    printf("Enter integers to sum (0 to stop)\n");
    int input = 0;
    int sum = 0;
    do {
        printf("Enter integer:  ");
        /* no error checking since it's not feasible for the MIPS program */
        scanf("%d", &input);
        sum += input;
    } while (input != 0);
    printf("Sum = %d\n", sum);
    return 0;
}
```

   Notice in particular that no error checking is required. You are allowed — encouraged, really — to use the procedures in `subs.s` on the "sample programs" page to do the needed I/O.

2. (10 points) Here is a partial MIPS assembler program for the SPIM simulator that prompts for two integers and then computes and prints their greatest common divisor (GCD): gcd.s[1].

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS2321_2015spring/Homeworks/HW05/Problems/gcd.s`

Your mission is to complete it so that it actually does the computation, *without making any changes other than to add a `gcd` procedure.* You will likely lose points if you make other changes.

*ADDITION:* You will also lose points if you do not follow MIPS conventions for procedures — obtaining parameter values from `$a0` and `$a1`, returning a value in `$v0`, and saving/restoring any of the `$sN` registers you use.

Your program only needs to work if the inputs are both positive integers; you do not need to add anything to make sure this is true. A simple recursive algorithm for computing the GCD of two integers can be expressed in pseudocode thus:

```
gcd(a, b) {
    if (b == 0) return a;
    else return gcd(b, a%b);
}
```

Your program will only get full credit if you use this algorithm. Solutions using loops are possible but not in my opinion any simpler, and implementing using recursion is a better test of your understanding of the use of procedures. (If you wonder how to compute a remainder in MIPS assembler: The `div` instruction computes both quotient and remainder and puts them in special-purpose registers which you can access with the `mflo` (for the quotient) and `mfhi` (for the remainder) instructions.)