CS 3410 Homework 3 NetID_____ Date:_____
Due Tuesday, March 13st, 2012, 11:59pm Name:_____

1.

a) A common compiler optimization often takes often takes the code of a particular subroutine and embeds it in the calling function directly instead of calling the subroutine. Explain why this leads to higher performance.

b) The MIPS calling conventions include both callee-save and caller-save registers. Why don't we just make all registers either callee-save or caller-save?

2. Consider the following MIPS assembly code, run on a 5-stage pipeline:

```
        ADDI $s1, $zero, 0
        ADDI $s0, $zero, 0
        ADDI $s2, $zero, 0
LP1:    ADDI $s0, $s0, 1
        SLL $t2, $s0, 2
        ADD $t2, $t2, $s7
        LW $t0, -4($t2)
        ADD $s1, $s1, $t0
        SLT $t1, $s2, $t0
        BEQ $t1, $zero, BNT1
        NOP
        ADDI $s2, $t0, 0
BNT1:   BNE $t0, $zero, LP1
        NOP
```

   a) Describe what the registers $s0, $s1, $s2, and $s7 contain when this code has finished running. That is, describe in English what the code does to these registers; don't try to give specific values. For $s7, how is it used? That is, what common programming construct does $s7 represent?

b) Suppose we are using the 5-stage pipelined processor in Patterson and Hennessy. This code executes properly by handling all hazards. List all data and control hazards in this code, the line numbers on which they occur, any registers involved, and how they are handled. This includes specifying which stages are involved in any forwards, any stalls that occur, and what is done with any delay slots.

c) Rewrite this code so that it runs more quickly. The final result must be the same for all registers, and no new registers should be used. Our solution is 12 lines long, with 8 lines in the loop. +3 bonus points if you do as well as our solution; +5 if you beat it!

3. Note that, although on the last assignment, we accepted answers from the unrevised 4th edition (Blue book), we will not this time. If you do not have that book, ask for the details on Piazza.

   a) Problem 2.36.1 (a) on page 214 of the Patterson and Hennessy book, revised 4th edition.

   b) Problem 2.36.4 (a) on page 214 of the Patterson and Hennessy book, revised 4th edition.

   c) Problem 2.37.1 (a) on page 215 of the Patterson and Hennessy book, revised 4th edition.

   d) Problem 2.37.3 (a) on page 215 of the Patterson and Hennessy book, revised 4th edition.

4. Euclid's algorithm is one of the oldest and most famous algorithms in existence. It finds the greatest common divisor of two positive integers, and works as follows:

   Given two positive integers num1 and num2, subtract the smaller number from the larger and replace the larger number with that difference. That is, if num1 > num2, set num1 = num1 – num2. Repeat this process until one number is zero. The remaining nonzero number is the greatest common divisor of the original two numbers.

   a) Suppose variables num1, num2, and val are given. Write a code segment in C or pseudocode that uses Euclid's algorithm to compute the greatest common divisor of num1 and num2 and stores the result in val.
      Note: If you are familiar with Euclid's algorithm, you may know that there is a common way to improve upon the algorithm described above. Do *not* use this optimization; implement Euclid's algorithm as described.

b) Suppose that num1 is stored in register $a0, num2 is stored in register $a1, and val is stored in register $v0. Translate your above code into MIPS assembly code. You must account for delay slots in any branches or jumps that you take, but you do not need to optimize the code in any way (so NOPs are fine).

Note: We are not looking for a piece of code that happens to produce the same result as the code above; we are looking for a *translation* of the code above. So each line of code above should be associated with one or more lines of MIPS code, and these should follow the same order on the page as the code above.