CS311 - Computer Architecture - Spring 2017
Homework 2 – 60 points
Due: Feb. 8

1. (5 points) Exercises 2.6.1, pg. 184. Try to do each with as few MIPS instructions and registers as possible.

2. (10 points) Exercise 2.7 (all 6 parts), pp. 185, 186. Instead of using 1000 for part b) for 2.7.4–2.7.6, using 1134 (the current number of undergraduate students at WC).

3. (10 points)

   (a) Determine the binary and hex representations of the following MIPS commands:

   ```
   a) lw $t3, -456($s4)
   b) sub $t0, $a0, $t5
   ```

   (b) Determine the binary and base-10 representation of the following hexidecimal numbers:

   ```
   a) 0x32F900FC
   b) 0x1611FFB4
   ```

   (c) What MIPS assembly language instructions would the above hexidecimal numbers translate to?

4. (5 points) Exercises 2.13.1, 2.13.3, pg. 191.

5. (10 points) Translate the following HLL code into MIPS assembly language

```
sum = 0;
for(i=1; i<=j; i++)
   sum += i;
```

   Assume `sum`, `i`, and `j` are in registers `$s0`, `$s1` and `$s2`, respectively. Determine the total number of instructions executed, in terms of the value of `j`.

6. (10 points) Consider the following C++ code and MIPS translation

```
     C++                                MIPS
while (a[i] == k)          Loop: sll  $t1, $s3, 2     # $t1 = 4i
   i++;                          add  $t1, $t1, $s6   # $t1 = address of a[i]
                                 lw   $t0, 0($t1)     # $t0 = a[i]
                                 bne  $t0, $s5, Exit  # exit loop if save[i] != k
                                 addi $s3, $s3, 1     # i++
                                 j    Loop
                           Exit: ...
```

   (a) Assuming the first value where `a[i] != k` is when `i=10`, determine the number of MIPS instructions executed by the loop if `i` starts at 0.

   (b) In the above code two branch/jump instructions are executed for each loop iteration (except the last). This turns out not to be optimal. Rewrite the above code so that only 1 branch/jump instruction is executed for each loop. Determine the total number of instructions executed using the conditions stated in part (a).

(over)

7. (10 points) The traditional way in which we swap two values is to use a temporary variable as follows:

```
int temp = a;
a = b;
b = temp;
```

   (a) Translate this code to MIPS, assuming `a` is in `$s0` and `b` is in `$s1`.

   (b) We can actually swap two values without using a temporary variable. Determine this method (it involves the same number of statements, but uses some arithmetic/logical expressions) and translate this into MIPS.

   (c) What is an advantage of this latter method?