Name: ______________________________ **RCS ID:** ____________ @rpi.edu

♣ ♣ ♣ CSCI 2500 — Computer Organization ♣ ♣ ♣
Fall 2018 Quiz 3 (October 12, 2018)

Please silence and put away all laptops, notes, books, phones, electronic devices, etc. This quiz is designed to take 25 minutes; therefore, for 50% extra time, the expected time is 38 minutes and 100% extra time is 50 minutes. Questions will not be answered except when there is a glaring mistake or ambiguity in a question. Please do your best to interpret and answer each question.

1. **(10 POINTS)** When a MIPS instruction is executed, which of the following is always true? Clearly circle the **best** answer.

   (a) The program counter is set to the branch target address

   (b) The program counter is incremented by 4 bytes

   (c) The pseudo-instruction is translated into an actual MIPS instruction

   (d) The fetch cycle fetches the target memory address

2. **(15 POINTS)** Given the three-input Boolean expression $\overline{(A+B)} \bullet \overline{(A+C)}$, what is a simpler equivalent form? Clearly circle the **best** answer.

(a) $A \bullet B \bullet C$	(c) $\overline{A} \bullet \overline{B} \bullet \overline{C}$	(e) $\overline{B} \bullet \overline{C}$
(b) $A + B + C$	(d) $\overline{A} + \overline{B} + \overline{C}$	(f) $\overline{B} + \overline{C}$

3. **(25 POINTS)** A *functionally complete set* is a minimal set of operators that can be used to represent any possible Boolean expression. We know from class that `AND`, `OR`, and `NOT` form a functionally complete set.

   (a) Do the two operators `NOR` and `NAND` form a functionally complete set (yes or no)?

   (b) If "yes," describe how. If "no," describe why not (i.e., describe what's missing).

4. **(50 POINTS)** Translate the given C function below into a MIPS procedure labeled `downcase`. More specifically, input register `$a0` will contain the memory address of string `s`. Be sure you write a complete procedure that uses the stack properly.

   You should use *Load Byte* (`lb`) and *Store Byte* (`sb`) instructions to deal with individual characters. You are allowed to use MIPS pseudo-instructions.

```
/* Change all uppercase letters in string s to lowercase, leaving all
 * other characters intact; return the number of "down-cased" characters
 *
 * ASCII: 'a' is 0x61; 'z' is 0x7a; 'A' is 0x41; 'Z' is 0x5a
 */
int downcase( char * s )
{
  int count;

  for ( count = 0 ; *s ; s++ )
  {
    if ( *s >= 'A' && *s <= 'Z' )
    {
      *s += 0x20;
      count++;
    }
  }

  return count;
}
```