CSCI 2500 — Computer Organization
Lab 09 (document version 1.1)

- This lab is due by the end of your lab session on Wednesday, November 6, 2019.
- This lab is to be completed **individually**. Do not share your code with anyone else.
- You **must** show your code and your solutions to a TA or mentor to receive credit for each checkpoint.
- Labs are available on Mondays before your lab sessions. Plan to start each lab early and ask questions during office hours, in the discussion forum on Submitty, and during your lab session.

This lab will consider 32-bit binary MIPS instructions. For this lab, we'll implement functions to decode the binary into its operational components. See the comments in file `lab09.c` for detail.

1. **Checkpoint 1:** For the first checkpoint, use C to finish implementing the `push_lsb_bit()` and `pop_lsb_bit()` functions. Review the `print_b32()` function, too, though do not change it.

   Begin by downloading the `lab09.c` code, which provides fill-in-the-blank skeletal code for these (and other) functions. Verify that the test cases given in `main()` provide correct results, but also add additional test cases.

2. **Checkpoint 2:** For the second checkpoint, continue to add to the `lab09.c` code by implementing the `isolate_bits()` function. Uncomment the test cases in `main()` to test your `isolate_bits()` function. Further, add additional test cases to be sure your function is correct.

3. **Checkpoint 3:** For the third checkpoint, complete the `lab09.c` code by implementing the `decode_inst()` function. As part of this function, display the decoded instruction opcode using the format shown in the commented code. This function must also display the ALU control lines, when applicable.

   You only need to implement decoding for the "simple" instruction set we looked at in Chapter 4. See the second page of `csci2500-f19-ch04a-slides.pdf`.

   Uncomment the test cases in `main()` to test your `decode_inst()` function. Further, add additional test cases to be sure your function is correct.

   As a hint, in addition to the textbook, consider using `spim` to see what the decoded `opcode` codes, `funct` codes, etc. are for various instructions.