# CSCI 2500 — Computer Organization
# Lab 05 (document version 1.2)

- This lab is due by the end of your lab session on Wednesday, October 3, 2018.
- 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 by 6:00PM 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.

1. **Checkpoint 1:** For the first checkpoint, write MIPS code to determine whether a given unsigned integer in register `$a0` is even or odd. Implement this as a procedure (i.e., function) labeled `isodd`. This procedure must display either "`EVEN`" or "`ODD`" via the appropriate `syscall`, then return `0` if even or `1` if odd. Note that to return a value to the caller, use register `$v0`, then view the contents of this register via the `print_all_regs` or `print` command in SPIM.

2. **Checkpoint 2:** For the second checkpoint, write a MIPS procedure to extract and display each of the four individual bytes of a given unsigned 32-bit value in register `$a0`. You must implement two working versions to earn full credit for this checkpoint; in the first version, you are only allowed to use `sll` and `srl` instructions, then in the second version, you must minimize the number of `sll` and `srl` instructions that you use by first using `and`, `or`, and/or `nor` instructions.

   As an example, if register `$a0` contains unsigned integer `0x4a889cf1` (i.e., 1,250,467,057 in base 10), your procedure should output the following:

   ```
   byte #1: 74
   byte #2: 136
   byte #3: 156
   byte #4: 241
   ```

   Test your procedure by calling it from `main` with a variety of input values in `$a0`. Further, you are encouraged to write reusable procedures to make your code easier to follow and more maintainable.

3. **Checkpoint 3:** For the third checkpoint, write a MIPS procedure labeled `numodds` that counts the number of odd integers that appear in a given list of integers. Use register `$a0` to specify the base address of the array and register `$a1` to specify how many 32-bit unsigned integers are in the array. Finally, return the number of odd integers in register `$v0`.

   Reuse the procedure from the first checkpoint to implement your solution to this checkpoint. And to test your code, use the `.word` directive to store a sequence of values in memory, e.g.:

   ```
           .data
   list:   .word 12,100,101,5,123456789,18
   ```

   Use `print_all_regs` or `print` to see the result of your function call (i.e., by looking at the contents of register `$v0`).