CSCI 2500 — Computer Organization
Lab 03 (document version 1.0)
Fire up your MIPSes

- This lab is due by the end of your lab session on Wednesday, September 29, 2021.
- 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 and answer their questions to receive credit for each checkpoint.
- Labs are available on Mondays before your lab session. 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:**

   - Download a MIPS simulator for your system and get it installed and running. See Lecture 8 notes and the lecture itself for details.
   - After installing the simulator, download the `helloworld.s` code from Course Materials on Submitty, then run the code in the simulator. You may refer to Appendix A of our textbook and use this URL as a reference for how to write MIPS programs: `http://web.archive.org/web/20200208081035/http://students.cse.tamu.edu/pritam2309/csce350/reference/quick_ref_MIPS.html`. Another useful resource is the "MIPS Reference Data Card" from your textbook, also available under Course Materials on Submitty.
   - Modify the given `helloworld.s` code by changing the "Hello world" string to instead be "MIPS is awesome!" (and add a newline to the end of the output).

     Take the time to understand how this example program works. Be sure you understand how to set up the system call (i.e., `syscall`). In particular, what happens if you load register `$v0` with `1` instead of `4`? In other words, describe what happens when you change the first `main` line to:

     ```
     main:  li $v0, 1       # syscall 4 (print_str)
     ```

     What happens when you change the above value to `2`? To `3`? And so on?
   - Try and run the `load_store.s` code from the lecture as well as `array.s` from the examples that come with this lab. These examples along with Chapter 2 slides will be extremely helpful in completing the rest of this lab.

     After you're comfortable with reading, understanding, and running the above code, its now time to write some of your own.

2. **Checkpoint 2:** For the second checkpoint, you will translate the following block of high-level code into MIPS:

   ```
   {
     int x = 2;

     if ( x < 7 )
     {
       x -= 4;
     }
   }
   ```

   To accomplish this, associate variable `x` with register `$t0`. In other words, register `$t0` will hold the value of `x`. After you implement this logic in MIPS, use MIPS code to display result `x` using the following output format:

   ```
   x ($t0) equals <value-of-x>
   ```

   As an example, the above code should output the following:

   ```
   x ($t0) equals -2
   ```

   You will need to make use of the `print_string` and `print_int` system calls, as well as the Set Less Than (`slt`)/Set Less Than Immediate (`slti`) and Branch On Equal (`beq`) instructions.

   Test your MIPS code by changing the initial value of x (`$t0`) to `0`, `4`, `5`, `7`, `100`, etc.

3. **Checkpoint 3:** For the third checkpoint, we will implement the GCD algorithm using MIPS. The greatest common divisor (GCD) of two numbers can be determined using Euclidean algorithm (https://mathworld.wolfram.com/EuclideanAlgorithm.html). Implement the iterative algorithm to determine the GCD of various values. More specifically, in MIPS, implement the GCD algorithm that uses subtraction instead of the "mod" operator.

   You may find it easier to first implement the Euclidean algorithm in C and then translate your C code into MIPS.

   To complete this checkpoint, use system call `syscall` 5 (`read_int`) to read two integer values for your GCD code from keyboard into registers `$t0` and `$t1`. Make sure you use positive values less than `32768`.

   Finally, display the resulting GCD of the two given values. Use the format shown below:

   ```
   GCD( <first-value>, <second-value> ) is <GCD-result>
   ```

   As an example, if values `$t0` and `$t1` were entered as `45` and `54`, respectively, the console would contain:

   ```
   45
   54
   GCD(45, 54) is 9
   ```

   Try it again with inputs `54` and `45`; in other words, make sure it works if you swap the two inputs! Test your MIPS code with different input values.