CMPT350: Lab Demo 5

Learning bitwise manipulation in MIPS *1%*
Due: October 26, 17:00

An important part of assembly-level programming is understanding the importance, power, and usage of bitwise operators (AND, OR, NOT, etc). Understanding binary data representation and how and when to apply bitwise operators in problem contexts can lead to much more streamlined code. In this lab, we'll investigate how to efficiently talk about and modify the case of a provided letter character using bitwise operators.

Laboratory Procedure:

On the course Moodle page, look for the Demo 5 assignment in the Laboratories section. Inside is a premade skeleton MIPS file, `demo5.s`. Save this somewhere handy on your machine. You may use both the command line version, `spim`, and the GUI version, QtSpim to run your program. As well, you will need a text editor to edit your programs.

Assignment Requirements:

- Prompt the user for a letter, using the appropriate system call. (Assume the user always provides a valid character: either an uppercase or lowercase letter).
- Determine the case of the character from its binary representation using your choice of bitwise operators
- Copy the character to a new register and modify it to be the lowercase version of the original character, regardless of the case of the original character using your choice of bitwise operators
- Copy the original character to a new register and modify it to be the uppercase version of the original character, regardless of the case of the original character using your choice of bitwise operators
- Toggle the case of the original character (switch it from lowercase to uppercase, or uppercase to lowercase), without branching, using your choice of bitwise operators.

*** **NOTE:** Your code **must interpret and execute** within the **laboratory environment**. Failure to do so will result in a **mark of ZERO** for the program. ***

Creating a trace file

As part of all demo and laboratory submissions, you must submit what's called a trace file. This is a text file you'll create that in essence replicates what your terminal looks like when you demonstrate your programs. To assist with this, a tool called `script2` has been provided for you. To create a trace file, simply run `script2` in your terminal, and then proceed to use the terminal as normal to navigate to and run your programs. When you have demonstrated everything, press CTRL+D to signal that you are done, and `script2` will exit, creating a file called typescript in the directory you ran `script2` from. This is your trace file. **Please rename your tracefile to include your name, e.g. baker_typescript**

Submission:

Your programs should be named using your name and end in a .s extension. For example, BakerDemo5.s

When you feel your programs meet all of the above requirements, call the lab coordinator to demonstrate your program, and create a zip archive with your code submission and trace file(s) included. An easy way to do this is with the `zip` command. For example, if your solution file and trace file are in a folder called `baker`, then the command `zip -r baker.zip baker/` will create a zip file containing your submission files called baker.zip (Replace this with your name when creating your zip file). Upload this zip file to Moodle.

Hints:

- This demo does not require the use of functions, though you may use them if you wish.
- Think back to the ASCII table to consider both the decimal and binary representations of characters. How do uppercase 'A' and lowercase 'a' differ? Uppercase 'Z' and lowercase 'z'? 'M' and 'm'?
- Only one bit in the 8 bit ascii representation of a character needs to be written to/read from to talk about the case of the character (assuming the character is indeed a letter).
- Each task can be achieved with a single bitwise instruction, but it can be useful to use several to reason through their use more easily.

Example Output: