CSE 260M - Homework 6

Always show all work for full credit.

1. Convert the following RISC-V instructions to binary. Show all work and the final value in hexadecimal (Big-endian format: "Big End" of number, which starts with the most significant bit ("MSB": bit [31]) leftmost. This is how problems worked in class were shown):

```
1) sll x5, x7, x6
2) lw a0, 32(sp)
3) neg s1, s2          # Hint: Table B.7
```

2. Convert the following 32-bit hexadecimal values to RISC-V instructions (use the register names, like `a0`) and express any numeric values or labels in decimal

   Hints:

   - Review Figure B.1 (first figure in Appendix B). Notice that all the instruction formats have field (part) in common. That part can be used to determine the instruction's format.
   - Once the format is known, other parts and values can be identified.
   - `imm` values are signed and could be negative!
   - Some `imm` values are split up into parts in unusual orders and have to be re-assembled. Such `imm` values may have "unspecified" bits that should be assumed to be 0

```
1) 0x00A4F2B3
2) 0x00512423
3) 0xFE058CE3
```

3. Complete code in `numberstats.s` (repo link on the course page), which should find the minimum, maximum, and sum of elements in a given array. Suggested workflow:
   - Review the documentation on Venus in VSCode: https://marketplace.visualstudio.com/items?itemName=hm.riscv-venus
   - Write *and test* the code in a programming language that you know. This will help you identify the concepts and critical logic. Assembly language is much easier if you already have correct logic.
   - Once it works, make a new copy to add annotations (comments) to that will help you convert it to assembly language. Use concepts from assembly language, like replacing variables with register names. For example, replace "sum" with "a2" and leave comments that clearly show that every place "a2" is used it represents the sum.
   - You'll need "labels" for loops and if/else statements. Add in comments indicating where you may need to use labels and what the label will be. Labels are like variable names --- they should be descriptive.
   - On individual lines add comments describing instructions that you may need to use to accomplish comparable work in assembly language.
   - Review and run `numberstats.s` before you start modifying it. Make you should and shouldn't modify it and what is expected. The provided code returns 1, 2, and 3 for the min, max, and sum. Note that the 5th line of the file contains test data that your code will use. You can modify this line to try different test cases.
   - Start to add your code, but start small. Include just parts for the `sum` first (not min and max).
     i. Use the "Run" button to initially load your program. Always check the "Terminal" panel for error messages.
     ii. If there aren't any syntax errors, it's best to try "stepping" through your code one instruction . You can hover your mouse in the "gutter" on the right of your first line of code to add a "breakpoint". When you hit play the simulator will stop running on any lines with breakpoints. You can then use the "step" buttons to step through each line of code. You can examine the values in registers via the integer register section in the left panel.
   - Once sum works, revise work to add in the min, then the max.