CprE 381 Homework 1

[Note: This homework covers material on the construction and definition of ISAs and begins to get you introduced to MIPS. It also has you start to run MIPS programs through a simulator (MARS) that you will use going forward both in homework AND to help test your term project. A little extra time spent working with MARS now will pay off later.]

1. ISAs
    a. Identify the (concrete) benefits and drawbacks (at least one of each and three total) of establishing a hardware abstraction that uses simple, primitive instructions rather than implementing a high-level programming language directly.
    b. Identify if the following items ***impact*** the ISA, ABI (but not ISA), or micro-architecture (i.e., a specific implementation of the ISA including which functional units are used and how they are interconnected). Note that some of these may have multiple answers. Provide a brief justification (1 sentence is sufficient).
        i. Number of named registers
        ii. Number of cycles an instruction takes to execute
        iii. Whether or not immediate operands can be used directly in arithmetic instructions
        iv. Which register number is the stack pointer
        v. Which, if any, registers numbers produce constants (e.g., 0 or -1)
        vi. Which register numbers pass arguments to function calls
        vii. Which register numbers are temporary or saved
        viii. Addressable address range for memory operations
        ix. Type of adder used in the ALU
        x. Which instructions update the PC (remember the PC holds the address of the next instruction to execute)
        xi. Which bits of an instruction correspond to the opcode or an operand
        xii. Number of functional units (e.g., adders or multipliers) in the processor

2. MIPS
    a. Identify three products not mentioned in lecture videos that use a MIPS-based processor.
    b. MIPS does not have a **clear** instruction (**clr** dst). The intent of this command is to set the "dst" register to 0. Why would such a simple, basic instruction not be included in a RISC ISA? List three arithmetic or logical instructions or instruction sequences not mentioned during lecture that produce the same effect as a **clr** instruction.
    c. What does the following program do (give a description in English sentences)?

```
xor $1, $1, $2
xor $2, $1, $2
xor $1, $1, $2
```

d. Assume that variables `a`, `b`, `c`, and `d` are mapped to registers `$s0`, `$s1`, `$s2`, and `$s3`, respectively. Write a MIPS program that implements

$$a = (\text{floor}(b / 16) \ \& \ c) - d \ \% \ 4$$

using only **`add`**, **`sub`**, **`addi`**, **`and`**, **`andi`**, **`or`**, **`ori`**, **`sll`**, and **`srl`** instructions. Since `b`, `c`, and `d` must not be overwritten, use as many temporary registers (`$t0-$t9`) as needed.

3. MARS Introduction
   a. Read the introduction to MARS found at https://dpetersanderson.github.io/Help/MarsHelpIntro.html. Some of the specifics of MIPS that are referenced you may just be learning about (or haven't learned about at all in the case of syscalls) However, you should be able to answer: does MARS simulate the ABI, ISA, and/or microarchitecture of MIPS? Explain. *[We'll use MARS since a modified version of it is used for testing the MIPS processors you will design for your term project.]*
   b. Download the MARS simulator from your Canvas course. Load prob3.s into your simulator. Run three times with different inputs and report the result (and corresponding inputs). Looking at the code, what does this program do? (write some C code that *could* compile to this)
   c. Modify prob3.s to find the minimum of two array values and write the value back to a third array location. The minimum of two numbers, x and y, can be given by:

$$MIN(x, y) = (x + y - abs(x - y))/2$$

To find the *abs* in MIPS, you may use the **abs $t0, $t1** *pseudo-instruction* (sets $t0 to abs($t1)).

All three indices should be entered by the user. For this assignment you may assume that the user inputs correct values. Provide three test cases (inputs and observed output) showing your code works and describe how you verified correctness. Submit your new program as prob3_<NetID>.s.