HW1: Intro to MIPS and Number Representations

This assignment will help you become more comfortable coding in MIPS and using binary and hexadecimal numbers. This is an individual assignment. You may work with your classmates but your final answers should be your own.

Deliverables: A file named *hw1.txt* (or .doc or .rtf or .tex or any other file that I can open on my computer without having to download new software).

If you haven't yet gone through the intro MARS tutorial (link is on Moodle), do that first!

Complete the following and record results/answers when requested in your hw1 file. Please do not copy the questions themselves, just write your answers with the appropriate number/letters. You also do NOT need to show your work, you can find a binary calculator with a quick google search to check your answers, as well as check the result in MARS, so you will definitely end up with the correct value, however it's important to practice doing the addition by hand first, as you will almost certainly (ok definitely) have to do some simple binary and hex math on the 1st exam.

1. Given the following 2 32-bit hexadecimal numbers

```
0x20b36080
0x200cc00d
```

   a. Add the numbers by hand and record the result (also in hex).

   b. Convert the hexadecimal result to binary and record the value (it's easiest to read if you leave a space between each set of 4 digits).

   c. Write the result in decimal assuming that the original hex values are 32-bit unsigned integers.

   d. Would the decimal value be different if the original hex values were 32-bit signed integers instead? If so what would the value be?

   e. Use MARS to write a small program (a few lines) in assembly language that adds these 2 numbers using the *add* instruction and check the result againts the one you found when adding by hand? Make sure to look at the result in MARS as a decimal as well as hex or binary. Copy your program instructions here to your HW

2. Given the following 2 32-bit hexadecimal numbers

```
0x40b36080
0x400cc00d
```

Repeat steps a-d from above and record your answers.

a.
b.
c.
d.
e. Again add these 2 numbers in MARS using your program from question 1. This time you won't seem to get a result - scroll down if need be to the bottom of the *Mars Messages* window to see what MARS has to say about what happened. Why did MARS have a problem?

f. Run your program again only change the *add* instruction to *addu* instead. What result does MARS show now? What does the "u" stand for in the *addu* instruction?

3. Given the following 2 32-bit hexadecimal numbers

```
0x90b36080
0x900cc00d
```

a. Add the numbers by hand and record the complete result with as many bits as you need.

b. Add these 2 numbers using the *addu* instruction in MARS. What is the result? Explain the difference in how these 2 add instructions (*add* vs *addu*) handle **overflow** (when the result of a calculation is too large to fit in the location used to store it). Why do you think the instructions are designed this way? (this can be pure speculation, no wrong answers as long as you try)