CprE 381 Homework 3

[Note: This homework gives you more practice with the MIPS assembly language. When you are asked to assemble a program, you can try running it on MARS to confirm it works. However, make sure you have it running in Bare Machine configuration so that your machine code matches the ISA.]

1. MIPS Machine Code
    a. The following instruction (until or unt) is not included in the MIPS instruction set:

    ```
    unt $t0, $t1, immediate
    # The first operand is rt, the second is rs
    # if (R[rt] != SignExtImm)
    #   R[rt] = M[R[rs]], R[rs] = R[rs] + 1, PC=PC
    ```

    If this instruction were to be implemented in the MIPS instruction set, what is the most appropriate instruction format? Explain why. Provide a sequence of MIPS instructions that performs the same operation. Ungraded, but exam-worthy: Postulate why this instruction wasn't included the MIPS ISA (there is a general reason and at least one very specific reason).

    b. Translate the following MIPS assembly into machine code providing the following for each instruction. First, identify the instruction's format. Second, provide the decimal value for each instruction field. Third, provide the hex encoding of the entire instruction. Assume `Begin` is at `0x00400000` (start of the text/code segment in the default memory configuration of MARS).

    ```
         addi  $t5, $zero, 1195
         j cond
    loop:
         sra   $t5, $t5, 1
         andi  $t6, $t5, 0x003C
         add   $t9, $a0, $t6
         sw    $t5, 4($t9)
    cond:
         bne   $t5, $zero, loop
         jr    $s7
    ```

    c. (**Optional**—not for credit, but useful for exam and project prep) We've discussed in class and on Canvas that you cannot load an arbitrary 32 bit integer (e.g., `0xFEED3210`) using a single instruction. Look up the **`lui`** instruction (e.g., on the green sheet from your textbook) and provide a two-instruction sequence that loads `0xFEED3210` into `$t0`. Then, assuming that **`lui`** is not supported by the ISA, provide a valid three-instruction sequence that loads `0xFEED3210` into `$t0`. Translate these into MIPS machine code providing the same steps as part 1.b.

2. MIPS Programming with SIMD *[I suggest you actually simulate these programs using the provided version of MARS to confirm that they work.]*
   a. Write a MIPS procedure that performs the 1D sum of absolute differences (SAD) of two byte arrays:
      ```
      int sad(char *array1, char *array2, int len) {
        char sum = 0;
        for (int i=0; i<len; i++) {
          sum += sat8(abs(array1[i]-array2[i]));
          // The result of abs(…) becomes max unsigned byte
          // if it were to exceed the capacity of a byte
        }
        return sum;
      }
      ```
      The 2D version of this problem is an important and widely-used computational kernel for applications such as video compression (e.g., mp4 motion estimation between frames). You should begin the procedure with a label and end it with **`jr`** `$ra`. You can assume the base address of the arrays are in `$a0` and `$a1` and the size of the first (and guaranteed shorter) array is in `$a2`. The result will go in the lowest byte of `$v0`. If you were to overflow a byte, you should simply return the maximum unsigned value you could store in a byte (this is called saturating addition). For example, if the two arrays were {0, 1, 2, 3} and {0, 0, 0, 0}, the result would be 6.
   b. Write a MIPS program that calls the procedure from 2.a three times with three unique input sets (different from the ones in 2.a) and prints the corresponding output.*[For example of printing outputs, look at prob3.s from HW1.]*
   c. Modify your MIPS code from 2a to use the quad byte MIPS DSP instructions presented in class. Please verify that your test cases still work. Note that the goal is to reduce the number of dynamically executed instructions, but not necessarily the number of static instructions. You can view the quick reference document (similar to the "Green Sheet") at https://s3-eu-west-1.amazonaws.com/downloads-mips/documents/MD00566-2B-MIPSDSP-QRC-01.00.pdf and the more complete documentation at https://s3-eu-west-1.amazonaws.com/downloads-mips/documents/MD00374-2B-MIPS32DSP-AFP-03.01.pdf if you are interested. *[I've included a modified version of MARS that includes the following instructions from MIPS DSP: **absq_s.qb**, **`addu.qb`**, **`addu_s.qb`**, **`raddu.w.qb`**, **`repl.qb`**, and **`replv.qb`**, **`subu.qb`**]*
   d. How many instructions (i.e., dynamic instructions) were executed in your programs? Show your calculations. *[MARS has a tool that can count instructions, which I suggest you use to verify your hand calculations.]*