Homework #1 **Due: 9/9**

Questions regarding instructions are referring to the MIPS32 instruction set.
The following apply to this homework:

- For those questions requesting MIPS code, minimize the number of instructions. (This is a challenge, but is not required for full credit.)
- Where indicated, present all binary in 32 bits (arranged in 8 groups of four bits each) and all hexadecimal numbers in 8 hex digits. (This helps both the grader and you.)
- Watch out for side effects: Don't modify (input) variables and registers that the problem doesn't allow you to modify.
- Show your work: Just writing down the answers, even if they are correct, is not enough and partial credit will depend on it. Attach worksheets, if any.

Reminder: When showing assembler language or doing computations with large numbers of digits in any base, quadrille ("graph") paper is your friend.

1. [18 points]

   Consider the following MIPS assembler:

   ```
   add $t0, $s0, $s2
   ori $t0, $t0, 0xff
   sub $s1, $t0, $s3
   ```

   Convert the above assembler to a single C assignment statement. Assume `int` variables `a`, `b`, `c`, and `d` are in registers `$s0` - `$s3`, respectively.

2. [22 points] Consider the following C code:

   ```
   a[i+2] = a[i-1] + b[4];
   ```

   Express this in commented MIPS assembler. Assume `$s0` contains the address of the `int` array `a`, `$s1` contains the address of the `int` array `b`, and `$s2` contains the `int` variable `i`. Use the `$t*` registers for temporary storage, if you need any.

   This can be done in 6 instructions.

3. [24 points] The following C code will determine whether an integer `i` is a power of 2 or not:

   ```
   (i > 0) && ((i & (i - 1)) == 0)
   ```

   If it's 1, `i` is a power of 2. If it's 0, `i` is not a power of 2.

   Express this in commented MIPS assembler. Assume that `i` is in register `$a0`. Use `$t*` registers for temporary storage. Use labels if you need to skip any instructions. Leave the result (1 or 0) in register `$v0`.

   This can be done in 6 instructions.

4. [18 points] Fill in the empty cells of the following table, converting the hexadecimal numbers first to binary and then to the corresponding MIPS instruction. Also indicate the format of the instruction.

hexadecimal	**binary (32 bits)**	**MIPS instruction**	**format**
0x02108024			
0x08000015			
0x12400008			

5. [18 points] Fill in the empty cells of the following table, converting the MIPS instructions to binary and then to hexadecimal

MIPS instruction	**binary (32 bits)**	**hexadecimal (8 hex digits)**
`add $t4, $s0, $t7`		
`sw $s3, 4($t0)`		
`bne $t0, $t1, 35`		