(10 pts) Exercise 2-8

(See number discussion in Chapter 2. For hex help, see http://www.danbbs.dk/~erikoest/hex.htm)
• What binary number does this hexadecimal number represent: $7ffe\text{_hex}$?

• What decimal number does it represent?

(5 pts) Exercise 2-9

• What hexadecimal number does this binary number represent: $1100\ 1010\ 1111\ 1110\ 1111\ 1010\ 1100\ 1110_{two}$
(5 pts) Exercise 2-11

- What is the MIPS assembly code for the following:
  
  ```
  if (g != j)  h = g - h;
  else         h = g + h;
  ```

  Variables f to j are assigned to registers $s0$ to $s4$

(5 pts) Exercise 2-12

- What is the MIPS assembly code for the following:

  ```
  if (j == h)  g = i + j;
  ```

  Variables f to j are assigned to registers $s0$ to $s4$
Exercise 2-13

What is the MIPS assembly code for the following:
\[
\text{\texttt{\textit{if ( (j == h) && (f != i) ) \texttt{\textbackslash{} g = i + j;}}}}
\]

Variables f to j are assigned to registers $s0$ to $s4$

Exercise 2-14

What is the MIPS assembly code for the following:
\[
\text{\texttt{\textit{if ( ( (g != h) && (f == i) ) ||}}}
\]
\[
\text{\texttt{\textit{(g == i) \texttt{\textbackslash{} g = i + j;}}}}
\]

Variables f to j are assigned to registers $s0$ to $s4$
(20 pts) Exercise 2-18: Pseudo-instructions

- Below you will see several problems of the form:
  - `li $t1, 0x7`  
    # $t1 = 7
- This is an example pseudo-instruction, with its meaning given as a comment. The instruction should load '0x7' into $t1. The constant is also called an 'immediate' – 'li' stands for 'load immediate'
- Constants beginning with '0x' are in hex. Thus, 0x7 = decimal 7. 0x17 = decimal 23 (16 + 7). Recall that each hex 'digit' is 4 bits, and that you can you can only use up to 16 bits for a simple constant.
- Each problem is separate – they don’t build on each other
- Your job is to write the real MIPS instruction (or sequence of instructions) that the compiler would produce for each given pseudo-instruction. For instance, the solution for the above pseudo-instruction is:
  - `addi $t1, $zero, 0x7`
- To make your answers simpler, you may make use of the 'li' pseudo-instruction where helpful (except when defining li itself), and may assume that it can handle constants up to 32 bits in size

- `clear $t0`  
  # $t0 = 0

- `beq $t1, 0x9, L`  
  # if ($t1 == 0x9) go to L

- `beq $t2, 0x12123434, L`  
  # if ($t2 == 0x12123434) goto L

- `li $t2, 0x12123434`  
  # $t2 = 0x12123434

- `bge $t5, $t3, L`  
  # if ($t5 >= $t3) go to L

- `lw $t5, 0x12123434($t2)`  
  # $t5 = Memory[$t2+0x12123434]