Assignment 2

Due: 9:00am, Tue Jan 30th, 2018

Note: Make reasonable assumptions where necessary and clearly state them. Feel free to discuss problems with classmates, but the only written material that you may consult while writing your solutions are the textbook and lecture slides/videos. Solutions should be uploaded as a single pdf file on Canvas. **Show your solution steps** so you receive partial credit for incorrect answers and we know you have understood the material. Don't just show us the final answer.

1. Annotate the following MIPS instructions to indicate source registers and destination registers. A source register is read during the instruction's execution, while a destination register is written during the instruction's execution. **(12 points)**
   1. add $s1, $t2, $t3
   2. lw $s3, 8($gp)
   3. sw $s4, 12($s5)
   4. addi $s1, $zero, 100
   5. bne $t1, $t2, else
   6. add $s1, $s1, $s1
2. Consider a program that declares global integer variables x, y[10]. These variables are allocated starting at a base address of decimal 1000. All these variables have been initialized to zero. The base address 1000 has been placed in $gp. The program executes the following assembly instructions:
   ```
   lw $s1, 0($gp)
   addi $s1, $s1, 25
   sw $s1, 0($gp)
   lw $s2, 12($gp)
   add $s2, $s2, $s1
   sw $s2, 8($gp)
   sw $s2, 12($gp)
   ```
   1. What are the memory addresses of variables x, y[0], and y[1]? **(15 points)**
   2. What are the values of variables x, y[0], y[1], and y[2] at the end of the program? **(20 points)**
3. Express the following decimal number in binary and hexadecimal forms: 146. **(6 points)**
4. Express the following binary number in decimal and hexadecimal forms: 1001100. **(6 points)**
5. Express the following hexadecimal number in decimal and binary forms: 0x6d. **(6 points)**
6. Write the MIPS assembly code that corresponds to the pseudo code below. Assume that the address for integer i is baseaddress+4 and the address for a[0] is baseaddress+8. Assume that the baseaddress is stored in $gp. The code initializes i to 0; it then iterates from i=0 to i=9, setting a[i] = 4i in each iteration. To make your code efficient, i must be loaded into a register at the start, and it must be updated in memory only after you've finished the for loop.
   ```
   for (i=0; i<10; i++)
   ```

```
    a[i] = 4*i;
```

(35 points)