6.004 Spring 2019 Tutorial Problems
L01 – Model of computing

Binary representation:

1. What is the 5-bit binary representation of the decimal number 21?

   10101

2. What is the hexadecimal representation for decimal 219 encoded as an 8-bit binary number?

   1101_1011 → 0xDB

3. What is the hexadecimal representation for decimal 51 encoded as a 6-bit binary number?

   110011 → 0x33

4. The hexadecimal representation for an 8-bit binary number is 0x9E. What is its decimal representation?

   158

5. What is the range of integers that can be represented with a single 8-bit quantity?

   0 - 255

6. Since the start of official pitching statistics in 1988, the highest number of pitches in a single game has been 172. Assuming that remains the upper bound on pitch count, how many bits would we need to record the pitch count for each game as a binary number?

   ceil(log2(172)) = 8

7. Compute the sum of these two 4-bit binary numbers. Express the result in hexadecimal.

```
 1101
+0110
 10011  == 0x13
```

Assembly Language:

LW	lw *rd, offset(rs1)*	Load Word	`reg[rd] <= mem[reg[rs1] + offset]`
SW	sw *rs2, offset(rs1)*	Store Word	`mem[reg[rs1] + offset] <= reg[rs2]`
ADDI	addi *rd, rs1, constant*	Add Immediate	`reg[rd] <= reg[rs1] + constant`
BEQ	beq *rs1, rs2, label*	Branch if =	`pc <= (reg[rs1] == reg[rs2]) ? label : pc + 4`
BNE	bne *rs1, rs2, label*	Branch if ≠	`pc <= (reg[rs1] != reg[rs2]) ? label : pc + 4`
BLT	blt *rs1, rs2, label*	Branch if <	`pc <= (reg[rs1] < reg[rs2]) ? label : pc + 4`
BGE	bge *rs1, rs2, label*	Branch if ≥	`pc <= (reg[rs1] >= reg[rs2]) ? label : pc + 4`

li *rd, constant*	Load Immediate	`reg[rd] <= constant`

Compile the following expressions to RISCV assembly using the instructions above. Assume a is stored at address 0x1000, b is stored at 0x1004, and c is stored at 0x1008.

1. a = b + c;

```
li a1, 0x1000     // actually lui a1, 1
lw a2, 4(a1)      // a2 = b
lw a3, 8(a1)      // a3 = c
add a3, a2, a3    // a3 = b + c
sw a3, 0(a1)      // store a3 into a
```

2. if (a > b) c = 17;

```
        li a1, 0x1000          // actually lui a1, 1
        lw a2, 0(a1)           // a2 = a
        lw a3, 4(a1)           // a3 = b
        bge a3, a2, end        // branch to end if a <=b (or b >=a)

        li a4, 17              // actually just addi a4, x0, 17
        sw a4, 8(a1)           // c = 17

end:
```

3. sum = 0;
   for (i = 0; i < 10; i = i+1)
      sum += i;

```
        li a1, 0               // a1 = 0 (sum)   or addi a1, x0, 0, since x0 is hardwired
to 0
        li a2, 0               // a2 = 0 (i)       or addi a2, x0, 0
        li a3, 10              // a3 = 10          or addi a3, x0, 10
loop:
        add a1, a1, a2         // a1 = a1 + a2 or sum = sum + i
        addi a2, a2, 1         // i = i+1
        blt a2, a3, loop       // if i < 10, branch to beginning of loop body
```