Homework 1

Question 1

For each of the 10-bit numbers specified by hexadecimal digits below, write down 1) the 10 bits in the number, 2) its value in decimal when the bits are interpreted as 10-bit unsigned binary numbers, and 3) its value in decimal when the bits are interpreted as 10-bit two's complement numbers.
a. 0x 1AB, **0110101011**, Decimal Signed **427**, Decimal Complement (1001010100) 597 **427**
b. 0x 2CD, **1011001101**, Decimal Signed **717**, Decimal Complement (0100110011) **-307**

Question 2

Convert the following decimal numbers to 8-bit 2's complement numbers, and then represent 8 bits with two hexadecimal digits.
a. Decimal 97, 2's complement **01100001**(**61**), complement of 10011110, 9E
b. -100, 2's complement 1**10011100**(**9C**), complement of 00011011, 1B

Question 3

For each instruction in the table, write 8 hexadecimal digits that represent the 32 bits in the destination register after the instruction is executed. Assume s0 is 0x98AB3C6A, s1 is 0x20503666.

1. 0x**B8FB72D0**
2. 0x**00003462**
3. 0x**B8FB3E6E**
4. 0x**B8FB0A0C**
5. 0x**98AB3E7A**
6. 0x**98AB3C60 INCORRECT**
7. 0x**B3C6A000**
8. 0x**FF98AB3C**

Question 4

3 instructions

```
sll s1, 3 #Shift left by 3 (equivalent to multiplying by 8=2^3)
addi t0, s1, 0 #store 8x in t0
add s1, s1, t0 # add 8x to 8x
add s1, s1, t0 # add 8x to 16x making 24x
```

Question 5:

if and is 0, then the beq will send to skip, meaning lines

```
loop:
    and t1, s0, t0 #
    beq t1, x0, skip #
    addi s1, s1, 1
skip:
    slli t0, t0, 1 #
    bne t0 x0, loop #
```

would run, totalling 4 lines.
The number of executed instructions does depend on the number of 1's in the number. For an input of all 0's, the program would run 130 times, and every 1 adds an extra instruction to increment the counter. This instruction set does not take into account the placement of the 1's, as the mask goes over every bit regardless of where they are placed or if there are only 0's left.
5a. **4**
5apt2. **146**

5b. There are many ways to compute Hamming weight. We could test the most significant bit (bit 31) of s0. For example, extract bit 31 with an AND instruction, and compare it with 0. This is similar to the method in the code given. However, we can save one instruction. If we treat s0 as a 2's complement number, s0 is less than 0 if since the binary unsigned and only if bit 31 in s0 is 1. Using this method, write RISC-V instructions to compute the Hamming weight of s0. Explain your method in comments. We can start with the following two instructions. How many instructions are executed if s0 is 0xFF00FF00?

```
addi s1, x0, 0 # s1 = 0
add t0, x0, s0 # make a copy so s0 is not changed
```

5b **3**
5b **90**

Question 6

```
lui s5, 0x55AABB33
OuterFor:
  bge s2, s1, Exit #if i>=a exit
InnerFor:
  bge s2, s3, ExitInner # if j >= i
  add t1, s3, s5 # Add j(s3) to 0x55AABB33(s5)
  xor s4, s4, t1 # xor r with temp storing (j+0x55AABB33)
  addi s3, s3, 1 # j++
ExitInner:
```

```
  addi s2, s2, 1 # i++
  beq x0, x0, OuterFor # Return to Outer for
Exit:
  # exit loop
```

This has 12 instructions, and I believe is the correct solution to number 6