Question 1 (25 points): You have been hired by Proyota, a manufacturer of embedded processors for cars. A new 8-bit processor is being designed and you need to help answer some questions about the processor. This processor has eight 8-bit registers named `R0`, `R1`, ..., `R7`, and also operates with an 8-bit word. Answer the following questions:

a. (**10 points**) In the table below indicate the values, in the specified forms, that can be stored in an 8-bit register

Description	Binary	Hexadecimal	Decimal
Largest unsigned integer	1111 1111	0xFF	$2^8 - 1 = 255$
Most positive 2-complement integer	0111 1111	0x7F	$2^7 - 1 = 127$
Most negative 2-complement integer	1000 0000	0x80	$-2^7 = -128$

Assume that the format for arithmetic instructions in this processor is as follows:

```
add   Ra, Rb, Rc                # Ra <-- Rb + Rc
sub   Ra, Rb, Rc                # Ra <-- Rb - Rc
```

Assume that the following values (given in binary) are stored in registers: `R1 = 0100 0011`, `R2 = 0100 0000`, and `R3 = 0100 0001`.

b. (**15 points**) What is the result, expressed in decimal, produced by the following sequence of instructions? Is it correct? If not, why not? If the code does not produce the expected result, is there a way to rewrite it to produce correct result? If yes, write the code that performs the correct operation.

```
add   R4, R2, R3
sub   R5, R1, R4
```

```
  R2 = 0100 0000
+ R3 = 0100 0001
-----------------
  R4 = 1000 0001   <-- overflow
```

Because of the overblow above, now we are subtracting a negative number from R1. This is performed as the addition of the complement of the number plus 1:

```
  R1 = 0100 0011
   +   0111 1110 = complement of R4
   +         1
-----------------
  R5 = 1100 0010  = -62
```

The second operation also results in overflow, and the effect of the two overflows actually cancel each other. Thus the result computed is actually the expected result: $67 - (64 + 65) = 67 - 129 = -62$.

However, it is best to avoid the overflows and to rewrite the code as shown below:

```
sub     R4, R1, R2      # subtracting two positive numbers, cannot generate overflow
                        # result of the subtraction above is 67 - 64 = +3
sub     R5, R4, R3      # again the two operands are positive, cannot generate overflow
```