Lab Exercise 2

In order to better understand bit representations of integers and floating-points, following exercise is designed. It is required to perform this exercise and submit your work to Blackboard by the end of the lab session. You are required to perform all operations using vi text editor.

1. Instructions

In this task, you are expected to implement two functions that perform bitwise operations and determine the bit representations of two floating point numbers. The functions that you have to implement are as follows.

- int bitParity8bit(int input)

  This function accepts an 8-bit unsigned number as an input. It returns 1 if the input number has an odd number of 1-bits, and returns 0 if the number of 1-bits is even. For example, bitParity8bit(7) = 1 because 7 is 0b111 in bits, and therefore, has 3 1-bits (odd). Another example is bitParity8bit(17) = 0 because 17 is 0b10001, and has 2 1-bits (even).

- int bitSum8bit(int input)

  This function accepts an 8-bit unsigned number as an input, and counts the number of 1-bits in that number. For example, bitSum8bit(7) = 3 and bitSum8bit(17) = 2.

Please note that you are not allowed to use loop statements in any of these functions.

In addition to writing these functions, you also have to determine the bit representations of 1.125 and -6.5 based on the 8-bit floating representation in the page 6 of lecture 6 slides, and write the values of the bit representations as 1-byte hexadecimal numbers. In this 8-bit representation, 1 bit is allocated for sign, 4 bits are for exponent, and 3 bits are for fraction.

2. Compilation and Correctness Checking

After finishing your implementation,you need to compile and check the correctness of your code. To compile the code, run ‘make install‘, and to check the correctness of the code, run ‘make test‘.