

CS2100: Computer Organisation
Tutorial #1: Number Systems and Codes
Answers to Selected Questions

This document is provided because some students might have missed the tutorial due to Chinese New Year holiday. Note that it is not a normal practice for us to release answers, hence please do not assume that answers for future tutorials will be released.

4. Question 16 in CS2100 AY2007/8 Semester exam paper is given below.

Given this code:

```
int n = 2147483640;
for (int i=1; i<=10; i++)
    n = n + 1;

System.out.println("n = " + n);
```

Given also that `int` type uses 32 bits, and that $2^{32} = 4294967296$. What is the output of the above code? Explain your output.

Answer:

The output is:

n = -2147483646

Here are the values of `n` in the 10 iterations:

2147483641, 2147483642, ..., 2147483647
-2147483648
-2147483647
-2147483646

Since `int` uses 32 bits, the largest positive integer is $2^{31} - 1 = 2147483647$, which is the value of `n` after the 7th iteration. In 2s complement representation, this is $(01111111111111111111111111111111)_{2s}$.

In the 8th iteration, the next value is $(10000000000000000000000000000000)_{2s}$, which is -2147483648.

6. If we generalize $(r - 1)$'s-complement to include fraction:

$$(r - 1)\text{'s complement of } N = r^n - r^{-m} - N$$

where n is the number of integer digits and m the number of fraction digits. (If there is no fraction digit, then $m = 0$ and the formula becomes $r^n - 1 - N$ as given in class.)

For example, the 1's complement of 011.01 is $(2^3 - 2^{-2}) - 011.01 = (1000 - 0.01) - 011.01 = 111.11 - 011.01 = 100.10$.

Perform the following binary subtractions of values represented in 1's complement representation by using addition. (Note: Recall that when dealing with complement representations, the two operands must have the same number of digits.) Check your answers by converting the operands and answers to their actual decimal values.

(a) $0101.11 - 010.0101$

(b) $010111.101 - 0111010.11$

Answers:

(a) **0011.0111_{1s}**

Check: $5.75 - 2.3125 = 3.4375$

(b) **1011100.110_{1s}**

Check: $23.625 - 58.75 = -35.125$

8. Given the following code C:

{ 0101010, 1001100, 0011001, 1110000, 0100101, 1000011, 0010110 }

(a) What is the **Hamming distance** of code C?

(b) What code word can you add into the code C yet retaining the same Hamming distance?

Answers:

(a) **4**

(b) **1111111**