Homework 2

Homework 2 is due on Wednesday, March 4, at the start of the lecture. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

Be sure to show your work on all your homework problems; otherwise, you risk not receiving credit.

It is recommended that you do not use a calculator (or online tools, etc.) when you do your homework. Remember that there will be no calculators allowed on exams.

Binary Representation and Arithmetic

1. Binary addition: unsigned and 2's complement

Consider the following 3 addition problems.

1. 1011 1010 + 0101 0101
2. 0110 1110 + 0110 1110
3. 1011 1101 + 0110 1011

a. Assume that the numbers are represented in **8-bit unsigned binary**. For each example, perform the addition *in binary (i*.e., *show the binary carries) and* give the decimal values of the operands and the sum. Does overflow occur?

Note: Give the decimal value of the 8-bit sum, even if there is overflow (i.e., even if the sum is incorrect).

b. Repeat part (a) assuming that the numbers are represented in **8-bit 2's complement.**

2. 2's complement subtraction

Do the following **8-bit 2's complemen**t subtractions. Show all steps in your solution, **including the translation to binary addition**. State if overflow occurs.

1. 0100 0101 - 0101 1110
2. 1100 0001 - 0110 0110
3. 0100 0110 - 1100 1011

3. Interpretation of Data Types

Give the value represented by string **x766B** if the data type encoded is the following:

Note: If it is not a valid representation, answer 'none' and explain.
Note: Your result should not be in binary.

1. Unsigned
2. 2's complement
3. single-precision IEEE 754 floating point
4. Extended (8-bit) ASCII string

4. Conversion from 2's complement: the positional weighting method

If $X = x_{n-1}\, x_{n-2}\, \ldots\, x_2\, x_1\, x_0$ is an n-bit 2's complement number, then the decimal value of X is $\mathbf{-x_{n-1}\, 2^{n-1} + (x_{n-2}\, 2^{n-2} + \ldots + x_2\, 2^2 + x_1\, 2^1 + x_0\, 2^0)}$

Example: The 2's complement number 1110 has decimal value $-2^3 + (1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0) = -8 + (4 + 2 + 0) = -2$

a. Use the *positional weighting method* to convert the following 2's complement numbers into their decimal representations.

1. 101100
2. 1110011
3. 1101101000

b. Repeat part (**a**) for these same 3 examples, using the 2's complement method.

5. Logical operations

a. Perform the following bitwise logical operations. Part 1 uses binary strings and the result should be in binary. Part 2 operates on hex strings and the result should also be in hex.

1. NOT(0010 OR (NOT(0100) AND 1101))

2. NOT(x3F OR (x8B AND x99))

b. Let a, b, c be Boolean variables; i.e., each of a, b, c can have the value 0 or 1.

- Q1 = NOT(NOT(a) AND b OR NOT(c))
- Q2 = (a OR NOT(b)) AND c

Corresponding to the above equations, complete the 3-variable (8 row) truth table given below.

a	b	c	NOT(a) AND b	a OR NOT(b)	Q1	Q2
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

What can you say about Boolean expressions for Q1 and Q2?

6. Conversion to floating point data type

Convert the following decimal numbers into their 32-bit floating point representation (IEEE single precision). For long binary strings, please insert spaces in order that the bits are in groups of 4. You may use a calculator to do the required multiplications, but you must show your work, not just the solution.

1. 15.4375
2. - 1.4

- **Hint 1:** Not all numbers have an exact representation using the floating point data type, so do not expect to always reach 0 after many multiplications.
- **Hint 2:** If your calculation has a repeating decimal, you may either truncate or round to get the last/rightmost/least significant bit.
- **Hint 3:** You may wish to convert your answer back to decimal to check the accuracy of your calculations.

7. Conversion from floating point data type

Convert the following 32-bit floating point numbers (IEEE single precision) into their decimal representation. For long binary strings, please insert spaces in order that the bits are in groups of 4. As always, show your work; you may use a calculator for your arithmetic (+,-,x,/) operations.

1. 1100 0100 1101 1001 0000 0000 0000 0000
2. 0011 1110 1011 0110 0000 0000 0000 0000