School of Computer Science
McGill University
Available On: Thursday, Sept. 21st, 2017.
Due Date: Thursday, October 5th, by 11:59 pm.
Hand in your solutions using myCourses, following the instructions below.
(late policy: 25% off per day late. As an example, the maximum you can get for an assignment that is one day late is 75%).

1 Number Representation and Boolean Algebra

Important! ***Organizing your answers for this question ***

- Write your solutions clearly on paper. You are free to either write your answers by hand, or to typeset them using a software package as long as they are written clearly and legibly. For typeset answers, don't use a font size less than 10pt. Note that you should demonstrate your working for the graders.
- If you have handwritten answers, scan them, and save the corresponding files into your answer-folder. Otherwise, for typeset answers, save the corresponding files in PDF or MS-Word formats into the answer-folder.
- Enter your numerical final answers in the answer-sheet that is provided for the Question 1 ("*question_1_answer_sheet.txt*"). You can find that file in *answer-folder*.
- After completing the "*question_1_answer_sheet.txt*" file make sure that your answer-folder contains a) the "*question_1_answer_sheet.txt*" file that just contains your final answers, and b) detailed answers (scanned papers, or typeset files) that show the steps you used to solve different parts of this question.

1.1 Number Representation (24 marks)

Convert the numbers below from the source base (left) to the destination base (right). Then enter your final answer into "*question_1_answer_sheet.txt*".
Q1.1.1) $(741)_{10} \rightarrow (?)_2$
Q1.1.2) $(741)_{10} \rightarrow (?)_{16}$
Q1.1.3) $(1.3515625)_{10} \rightarrow (?)_2$
Q1.1.4) $(1.3515625)_{10} \rightarrow (?)_{16}$
Q1.1.5) $(1001101)_2 \rightarrow (?)_{10}$
Q1.1.6) $(1001101)_2 \rightarrow (?)_{16}$
Q1.1.7) $(0.101011)_2 \rightarrow (?)_{10}$
Q1.1.8) $(0.101011)_2 \rightarrow (?)_{16}$

Q1.1.9) $(F00D)_{16} \rightarrow (?)_2$
Q1.1.10) $(F00D)_{16} \rightarrow (?)_{10}$
Q1.1.11) $(A.BED)_{16} \rightarrow (?)_2$
Q1.1.12) $(A.BED)_{16} \rightarrow (?)_{10}$

1.2 Floating Point Number Representation (16 marks)

Q1.2.1) Represent 1.00001 as an IEEE single precision floating point number in binary
Q1.2.2) Represent -0.32750702 as an IEEE single precision floating point number in binary

Whereas we haven't covered IEEE floating point number representation in class but I am going to provide an example in myCourses to show you how that works. You can then do a little bit of research on your own. Don't forget to show all your steps for this question.

2 Design of a 4-bit Adder-Subtractor (40 marks)

Important! ***Organizing your answers for this question ***
You should design your circuit in the provided template circuit file (*Four_Bits_Add_Sub.circ*). This file is located in *answer-folder* and you can open it with **Logisim-evolution** software.

2.1 Introduction

In class we saw a circuit diagram for a 1-bit full adder. In this assignment you will build a simple 4-bit adder-subtractor that supports two different functions (addition and subtraction). You must build your circuit in **Logisim-Evolution** using only the basic gates provided in the built-in library, specifically, AND, OR, NOT, XOR, and XNOR. You may set the properties of these gates such as changing the logical states of the inputs (e.g. negating them), the number of inputs, etc.

We have provided a starter project file for you to help you organize your solutions. You must implement your solution in that starter project by following rules that are mentioned below. You will also need to use wiring such as splitters to organize your implementation. To complete the objectives of this assignment, you must organize your solution into sub-circuits using the names and labels specified below (leave the main circuit empty).

- You will need to also edit the appearances of some sub-circuits to better organize your solution. But, be careful **not to change the sub-circuits' names, and input/output labels in the starter project!**
- You are free to create additional sub-circuits with custom appearances as you see fit and then use them in the starter sub-circuits. The sub-circuits for the main objectives, and in some cases the inputs and outputs, are already set up for you in the starter

logisim-evolution project file called "*Four_Bits_Add_Sub.circ*". Make sure that you are filling the starter project and its corresponding sub-circuits with correct designs, i.e., do not depart from the structure we have provided.

2.2 Warm up (10 points):

Implement a one-bit full adder in the "*Add_1Bit*" sub-circuit that takes A, B and C_{in} as single-bit inputs and produces the *Sum* and carry-out C_{out} functions. Note that the sub-circuit appearance was created for you in the starter code. The text labels in the are shortened forms of the input and output names defined in the circuit layout. Specifically, *Sum* is labeled with S for short.

2.3 Build a 4-bit adder/subtractor (30 points):

To do so you must fill in the starter sub-circuit ("*Add_Sub_4Bits*") with proper circuitry as detailed below:

You already have a 1-bit adder in the sub-circuit "*Add_1Bit*" of the project template file. Now, implement a 4-bit adder-subtractor in "*Add_Sub_4Bits*" by using four instances of "*Add_1Bit*" and complementary circuitry. (20 points)

There is a control input signal, "*Add_Sub*". Whenever it is asserted to '1', this circuit ("*Add_Sub_4Bits*") should perform 4-bit addition; otherwise when (*Add_Sub* = 0) the circuit should perform 4-bit subtraction (A-B) by using 2's complement methodology. For both cases your circuit should show the correct result on the output "R". Note that inputs (A, B) as well as the output R are represented as signed (2's complement) 4-bit numbers. Your implementation should be able to detect an overflow, if one occurs, and also to detect if the result of an operation has a value of zero (the template file provides two appropriate outputs).

3 WHAT TO HAND IN FOR THIS ASSIGNMENT

Everything should be handed in electronically on myCourses. Each student is to submit his or her own unique solution to these questions.

1. Zip JUST your *answer-folder*, rename it with your student ID number. For example, 260763964.zip
2. Submit this single compressed file on myCourses under Assignment#1.
3. Make sure that you submit a single file (the zipped file), not many files.

4 HOW IT WILL BE GRADED

- This assignment is worth a total of 80 points.
- The breakdown is stated on each question.
- Each question is graded proportionally. In other words, if your question is 50% correct you get 50% of the marks.
- For the first three days after the due date, we deduct 25% of the total possible marks, per day. After that no assignments are accepted.
- Any deviation from the submission instructions causes a 10% deduction.