King Fahd University of Petroleum and Minerals

###### Information and Computer Science Department

**ICS 102: Introduction to Computer Programming**

**Summer Semester 2018-2019 (Term 183)  
Term Project  
[Posted: Wednesday July 3rd 2019]  
[Due Date: Monday July 29th 2019 @ 8:00 AM]**

**Submission Guidelines:**

Submit a zipped file containing the Java source files

PLEASE DO NOT INCLUDE .class FILES IN YOUR SUBMISSION

The zipped file should be named as follows:

**PRJ\_XXXXXXXXX\_YourFamilyName.zip**

where:

XXXXXXXXX is your 9 digit KFUPM ID.

YourFamilyName is your family name

Submission should be made through your ICS 102 Blackboard course page under **Assignments** submission link.

**Important Notes:**

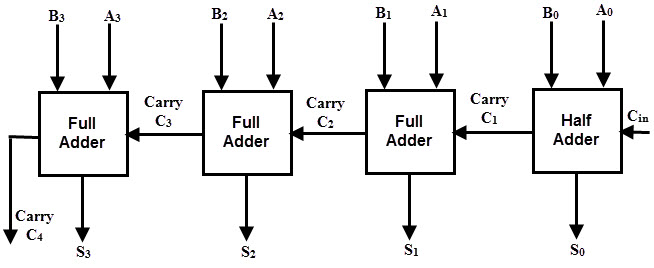
* **Cheating is taken seriously**. Any cheating attempt will result in an F grade in the course.
* **EACH STUDENT IS REQUIRED TO DO THE HOMEWORK ALONE**. COPYING FROM ANY SOURCE IS REGARDED AS CHEATING.
* **Submission link will be available until 9:00am for late submission without penalty.**
* **Submissions via email are not accepted and will be simply ignored**.
* Submission of the homework solution should be in a zipped filed with the format specified above. **Any different formatting/naming will result in reducing the total homework score by half!**
* **You must use proper indentation and meaningful variable names in your programs.**

**Problem Description**

Logic circuits are built using simple logic gates. An example is shown in Figure 1 which is called a *Full Adder* (FA) circuit that provides the sum of three bits A, B, and Cin. The operation of the circuit is summarized in a truth table (see Table 1). This table shows the outputs S and Cout for all possible combinations of the inputs, where S is known the sum bit and Cout is the carry bit. For example, the binary sum 1 + 1 + 1 results in S = 1 and Cout = 1. Note that an XOR gate is a simple circuit that takes two bits as input and generates a single bit output, where the output is 1 if and only if the two inputs are different**;** otherwise**,** the output is 0.

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated |  |
| Figure 1: Logic gates in the full adder logic circuit | Table 2: Truth table for the full adder logic circuit |

The addition operation of integer numbers occurring inside the processor is performed by an *adder* circuit. An adder circuit can be constructed using multiple FAs; an example is shown in Figure 2 for a 4-bit adder circuit. The adder generates a 4-bit output and a carry. The circuit in Figure 2 shows how to add the two binary numbers A3A2A1A0 and B3B2B1B0 to get S3S2S1S0 and the output carry C4. The input carry Cin should be set to 0.



Full Adder

Full Adder

Full Adder

Full Adder

Figure 2: Four-bit adder circuit

To add integer numbers, they are first converted into binary digits (bits), which are then added and the result is converted back to integer for display on the screen. For example, to compute 12 + 7, A = 12 and B = 7 and the addition is performed as follows:

Binary representation of 12 is 1 1 0 0

Binary representation of 7 is 0 1 1 1

Binary sum is 1  0 0  1  1 which is 19

The same circuit is used for subtraction of A – B by adding A to the 2's compliment of B (i.e. *inversion of bits in B then adding one*); in other words, adding the 1’s complement while making Cin = 1.

For the example, 12 – 7 =

Binary representation of 12 is 1 1 0 0

2’s complement of B is 1 0 0 1

Binary sum is 1  0  1  0  1

then invert the carry gives 00101 🡺 5

However, 7 - 12

Binary representation of 7 is 0 1 1 1

2’s complement of 12 is 0 1 0 0

Binary sum is 0  1  0  1  1

then invert the carry gives 1 if so, the results is negative of the 2’s complement of remaining bits. Thus it will – (0 1 0 0 1) 🡺 -5

Finally, the pseudocodes for conversion from integer to binary and vice versa are as follows:

*(a) convert an integer num to binary form*

Assume x=x0,x1,x2,x3,…..; //binary form

**for**(i=0; num !=0; i++)

xi = num % 2;

num /= 2;

*(b) convert binary to integer*

num=0; // integer value

**for**(i=0; i < SIZE; i++)

**if**(xi == 1)

num+= 2i;

*// another way*

int num=0;

**for**(i=0; i < SIZE; i++)

num+= xi \* 2i

**What to do:**

In this homework, it is required to simulate the operation of a *n*-bit binary adder and subtractor using adder to add/subtract two integer numbers in the range 0 – 15.

1. Create a utility class, named BinOperations, which has *static* methods for andGate(), orGate(), notGate(), xorGate(), bin2Int(), and int2Bin(). Note: the XOR operation is computed as follows:

1. Create a class called fullAdder which boolean member variables: A, B, Cin, S, Cout. The class should have methods to set and get A, B, Cin, and methods to get S and Cout. It should have another method run() to perform the full adder operation. This class is implemented using operations from the utility class.
2. Create a class called intAdder which has member variables: integer variables A, B, and S, boolean Cin and Cout. The class should have a constructor to create an adder of certain order n, e.g. n = 4 creates a 4-bit adder, i.e. it creates 4 fullAdders. It should have methods to set and get A, B and Cin, and methods to get S and Cout. It should also have a method add() to execute the addition operation by converting A and B into binary then use the fullAdders to compute the sum in binary and convert it back to integer. The conversion from integer to canary and vice versa is conducted using methods from the utility class.
3. Create a class Subtractor which has wraps an intAdder to compute the subtraction instead of addition. Besides set/get parameters, it should have a method sub() to execute the subtraction.
4. Create a driver Class SimulatorProject that has a main method to display a menu and prompts the user to select an operation then prompts the user for required operands if needed. Then execute the required operation.
5. Settings
6. Adder
7. Subtractor
8. Exit

In the beginning the user can only choose option 1. If the user chooses 1, the program prompts the user to choose the input stream either to be from keyboard or file, set the output stream to be to monitor or file. If the user choose file stream, should enter the file name. The program also reads n which is the maximum number of bits (order of the adder) and reads and validates non-negative integers A and B where each should be less than 2n. The input is read from the input stream set in option 1.

If the user chooses 2, the program creates an adder of order n, sets the values of A, B, Cin, and computes the sum. Based on the user settings, the output is written to the selected stream. The output shows A, B, and S in integer and binary forms.

If the user chooses 3, the program creates a subtractor of order n, sets the values of A, B, Cin, and computes the difference. Based on the user settings, the output is written to the selected stream. The output shows A, B, and D in integer and binary forms.

If the user chooses 4, the program terminates, showing the current settings and number of operations conducted of each type.

Any other key, the program shows an error and redisplay the menu.