Lab 3: Digital Logic, Data Types
Date Posted: September 26, 2022

1   Objectives

- Understanding data types

- Understanding logic gates

2   Resources

- Logic gates: `https://en.wikipedia.org/wiki/Logic_gate`

- Integer representation: `https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html`

- Floating point representation: `https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/`

3   Logic Gates

Logic gates can be drawn in two different styles (see resources link for an example). One way is shown in Figure 1, another way in Figures 2-5. For the truth tables in Figure 1, what are the outputs? Next, write a python script that contains **7** functions, one for each logic gate. The function should take in two boolean inputs and return one boolean output. (Except for the not gate, of course, which only has one input.)

4   Digital Logic Circuit

1. Trace through this circuit in Figure 2. Identify the logic gates and the outputs for the following set of inputs.

2. Using the functions you wrote in lab, implement this digital logic circuit in a python program. Your inputs should be a list containing four bits, and the outside should be a list of four bits.

5   Homework Questions

5.1   2 Bit Multiplier - 15 points

Figure 3 depicts a **2** bit multiplier diagram. Using the functions you wrote in lab, implement this two-bit multiplier in a python program. This takes as input two lists, with each element of the list an integer value of 0 or 1, and uses helper functions separately defined from the previous section. The output is a list of four integers, each integer of value 0 or 1. Note: The input and output lists all read from right to left, so smallest value digit of any of the lists is at index 0. Implement this in the function two_bit().

5.2   Integer Conversion - 30 points

Write a function called conversion() in Lab3.py that first asks the user choose to enter an integer or a binary number (int/bin).

If the user answers binary, then they must input a four bit number. If the user enters less than or more than four bits, prompt the user again. Then, your function should output the decimal value of the binary number assuming it is represented as sign magnitude, one's complement, and two's complement. For example, if the user inputs 1101, then the output should be "sign magnitude: -5, one's complement: -2, two's complement: -3."
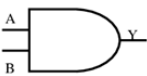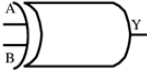
Basic Logic Gates

Logic	Schematic	Boolean Expression	Truth Table			English Expression
			A	**B**	**Y**	
AND		$A \cdot B = Y$	0	0		The only time the output is positive is when all the inputs are positive.
			0	1		
			1	0		
			1	1		
OR		$A + B = Y$	A	B	Y	The output will be positive when any one or all inputs are positive.
			0	0		
			0	1		
			1	0		
			1	1		
XOR		$A \oplus B = Y$	A	B	Y	The only time the output is positive is when the inputs are not the same.
			0	0		
			0	1		
			1	0		
			1	1		
NOT		$\bar{A} = Y$	A	Y		The output is the opposite of the input.
			0			
			1			
NAND		$\overline{A \cdot B} = Y$	A	B	Y	The output is positive provided all the inputs are not positive.
			0	0		
			0	1		
			1	0		
			1	1		
NOR		$\overline{A + B} = Y$	A	B	Y	The only time the output is positive is when all the inputs are negative.
			0	0		
			0	1		
			1	0		
			1	1		
XNOR		$\overline{A \oplus B} = Y$	A	B	Y	The only time the output is positive is when all the inputs are the same.
			0	0		
			0	1		
			1	0		
			1	1		

Figure 1: Logic Truth Tables

Figure 2: Logic Circuit
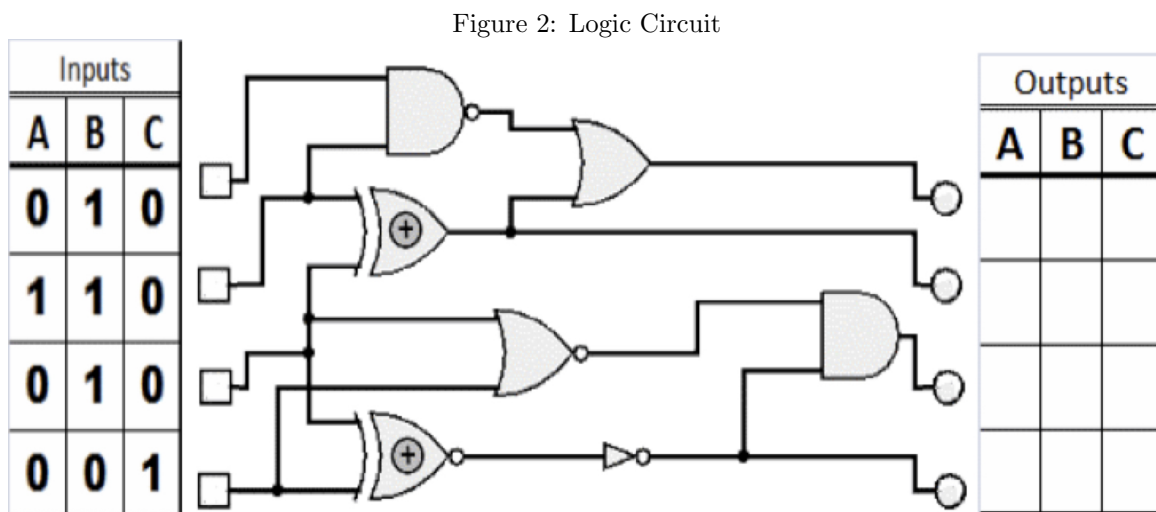
If the user answers integer, then then user must then input an integer, and your function will print that number in binary in three ways: sign magnitude, one's complement, and two's complement. If the user enters a number that is out of range for one or two of the outputs, then proceed with the conversion but output "out of range" for the respective outputs. If the integer is out of range for all three outputs, then keep prompting the user for a valid integer.

2

Figure 3: 2 Bit Multiplier
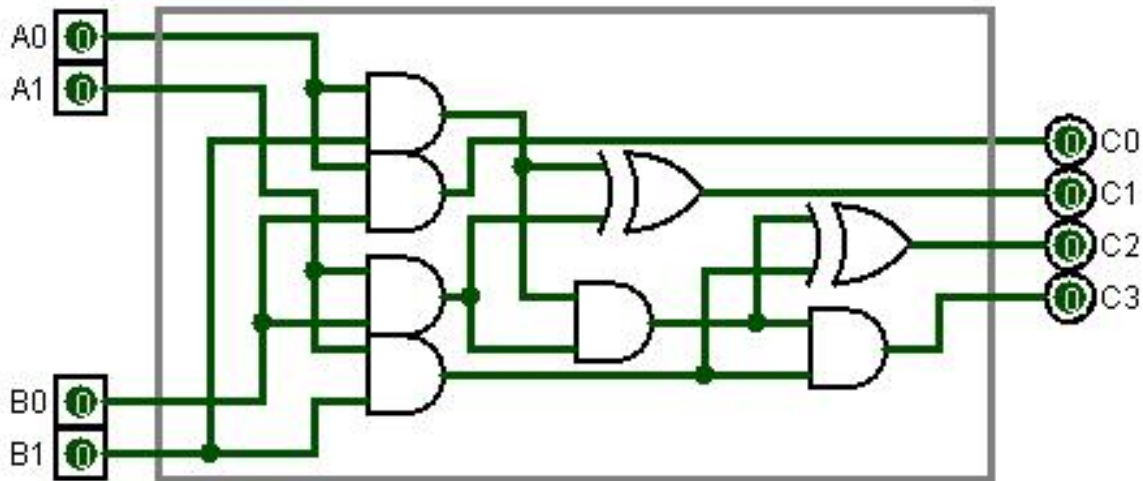
For this problem, you may not use built-in functions to convert between int, hex, or binary.

5.3 Two's to Dec - 10 points

Write a function in two2dec() in Lab3.py that asks a user for an 8 bit value represented in two's complement, and then converts this to decimal and prints this to the screen.

5.4 Floating Points: 20 points

In an 8 bit float, 1 bit is allocated for the sign, 3 bits are allocated for the exponent, and 4 bits are allocated for the mantissa.

- In dec2float(), write a function that asks the user for a decimal number, and prints the output in binary. If the user inputs a decimal number that is outside the range of a 8 bit float, then prompt the for another input.

- In float2dec(), write a function that asks the user for a 8 bit float, and then prints the decimal number to screen.

5.5 Hexadecimal Blackjack - 15 points

In blackjack, also known as "Twenty-One," the goal is to get as close to 21 points when compared to the dealer. See https://en.wikipedia.org/wiki/Blackjack or https://www.youtube.com/watch?v=qd5oc9hLrXg for full rules. Recall in class we played a version of Hexadecimal Blackjack in which a full deck of cards were values ranging from 0 to F instead of a standard deck which goes from 1 to Ace. In the class version, the code was very rudimentary in that the code kept running as long as we kept asking for a new card. In this homework problem you will be modifying the code to make the game more sophisticated. We will now be playing to 31. There are no face cards, so each card will be worth their hex value (i.e., a 2 is worth 2 points, an F is worth 15 points). To make it easier, you may assume that all inputs are valid, meaning that you don't have to account for a user typing in something unexpected.
Specifications:

- The game should initially ask if you would like to start a new game (y/n). If "n" then exit by saying "Goodbye!" If "y" then dealer should output "Your two cards are __ and __. Your hand sums to __" with cards randomly selected from a full deck containing four values each ranging from 0 to F. You may assume that the cards are replaced in the deck once they are drawn, meaning that there will always be a full deck to draw from in subsequent plays.

- The dealer outputs "Dealer's first card is __. Would you like a another card (y/n)?" If player keeps hitting "y" then keep printing out new cards as long as the player's hand does not exceed 31. At each step also print out the value of the sum of the hand. If the player's hand does exceed 31, then print out "Bust. Better luck next time."

- Otherwise, if the player chooses "n" and does not exceed 31, then next the dealer outputs: "Dealer's second card is __. Dealer's hand sums to __." If the dealer's hand is less than 28, then keep revealing a new card until the hand is greater than 28. If the dealer exceeds 31, then print out "You win!"

- If no player exceeds 31, then whoever is closer to 31 wins, and print out "__ wins!" If it is a tie, print out "No winner."

- Finally, ask if the player wants to play again. If so, then start the game over. If not, then print out "Goodbye!"

6 Submitting Homework to ELMS

Make sure to list all assumptions, and use comments to notate your code whenever possible. Clarity and style will matter. Your TF should be able to read your code and understand everything. You will be submitting two files.

1. A python directoryid_lab3.py file that contains all comments and code

2. A python directoryid_blackjack.py file that contains all comments and code for the blackjack question. Replace *directoryid* with your own directory ID in your lab files. Add your name and section number to the top of each file as a comment.