Logic Circuits Lab – Instructions

Overview

In the previous lab, you tested the operation of individual logic gates. Now, you will combine multiple logic gates into more complex logic circuits.

This laboratory has three required parts:

- Part 1: Buttons as Inputs
- Part 2: A Simple Logic Circuit
- Part 3: Designing your Own Logic Circuit

This laboratory also has three more *optional* parts if you want to stretch yourself. If you choose not to do the optional parts, you should at least read through them to learn the concepts.

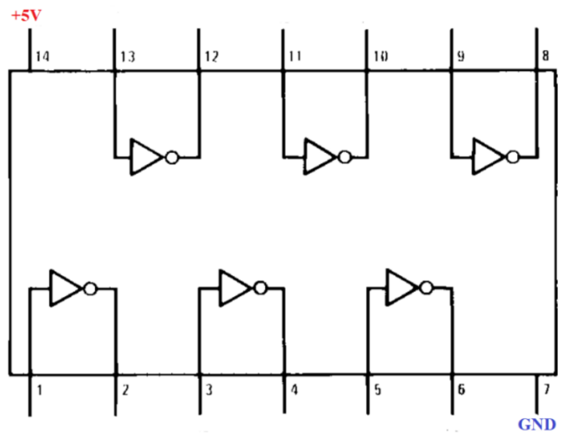
- Part 4: Three-Input Gates from Two-Input Gates
- Part 5: De Morgan Circuits
- Part 6: A Multiplexor Circuit

Required Equipment and Materials

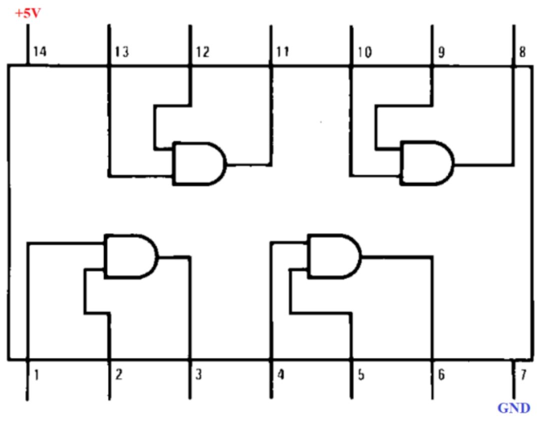
- The four logic chips you used from the last lab, which may still be in your breadboard:
    - 74LS04 NOT Gate Integrated Circuit (1)
    - 74LS08 AND Gate Integrated Circuit (1)
    - 74LS32 OR Gate Integrated Circuit (1)
    - 74LS86 XOR Gate Integrated Circuit (1)
- Buttons (3)
- Solderless breadboard (1)
- Red LED (1)
- 220 Ohm resistor (4)
- Male-to-male jumper wires
- To Power your Breadboard:
    - Arduino Uno (1)
    - USB cord (1)

Review: Data Sheets

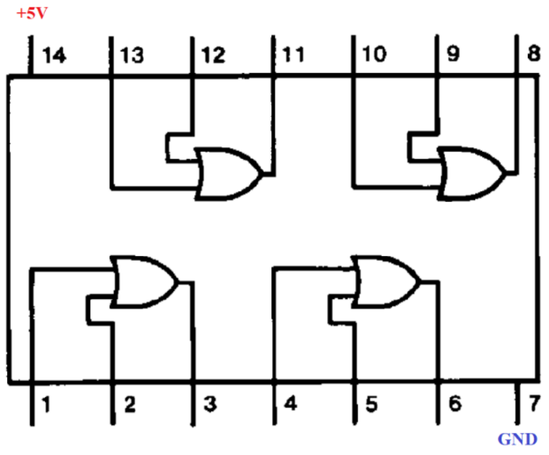
Review these diagrams from the datasheets to remember how the pins on each chip are connected to the logic gates inside that chip. You may refer back to these diagrams throughout the lab.



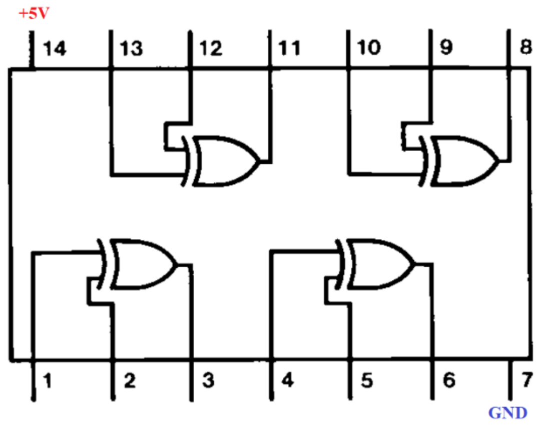
74xx04 NOT chip



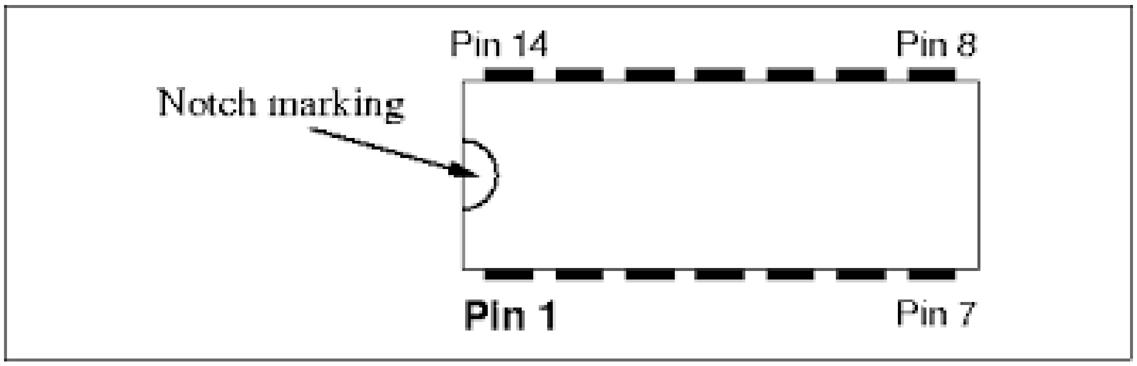
74xx08 AND chip



74xx32 OR chip



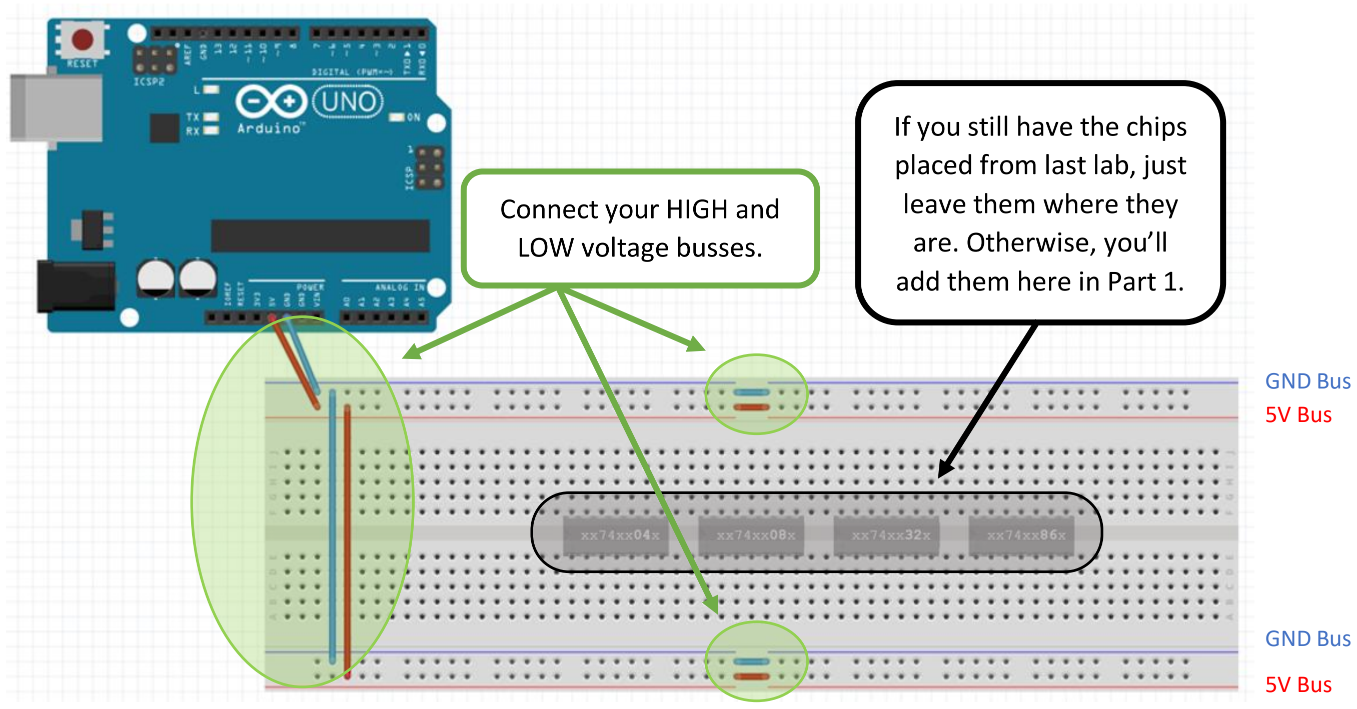
74xx86 XOR chip



Connect Power and Ground to the Breadboard

Connect power to your breadboard using the method from the last lab. Be sure to use jumper wires to connect the busses on the left half of the board to the busses on the right half of the board and the top busses to the bottom.

Note: If you left the chips in your breadboard from the last lab, you can keep them where they are for this lab.


Connect your HIGH and LOW voltage busses.
If you still have the chips placed from last lab, just leave them where they are. Otherwise, you'll add them here in Part 1.
GND Bus
5V Bus
GND Bus
5V Bus
xx74xx04x
xx74xx08x
xx74xx32x
xx74xx86x

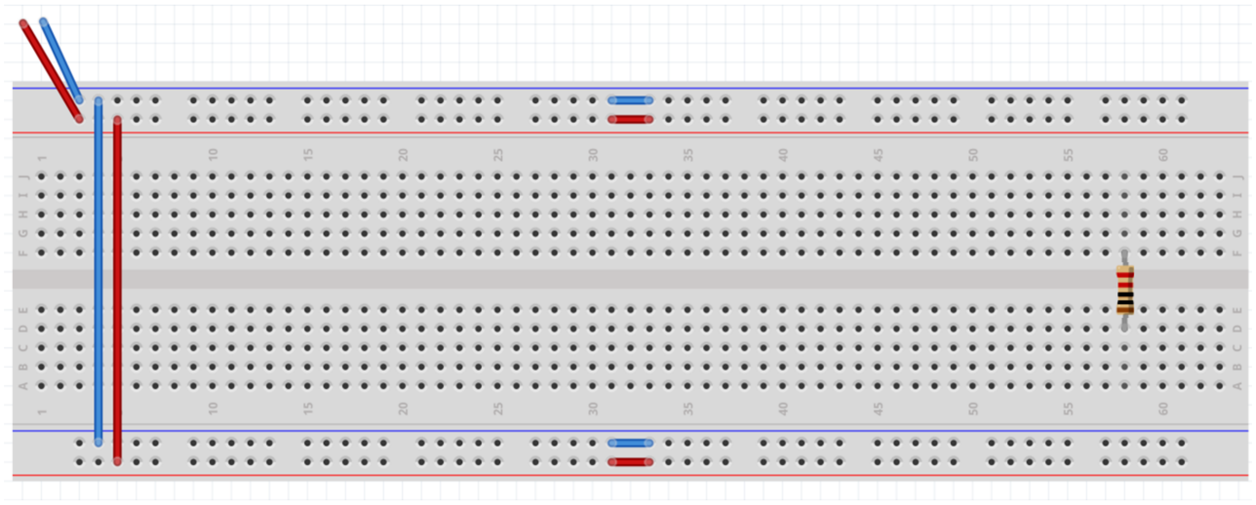
Throughout the lab, after you wire a circuit and are ready to test it, power the board by plugging your Arduino to your laptop. After you have finished testing a circuit, disconnect power by unplugging your Arduino from your laptop. Power should be disconnected whenever you are adjusting the wiring. Then, reconnect power whenever you are testing a circuit by reconnecting the Arduino to your laptop.

(Lab continues onto the next page.)

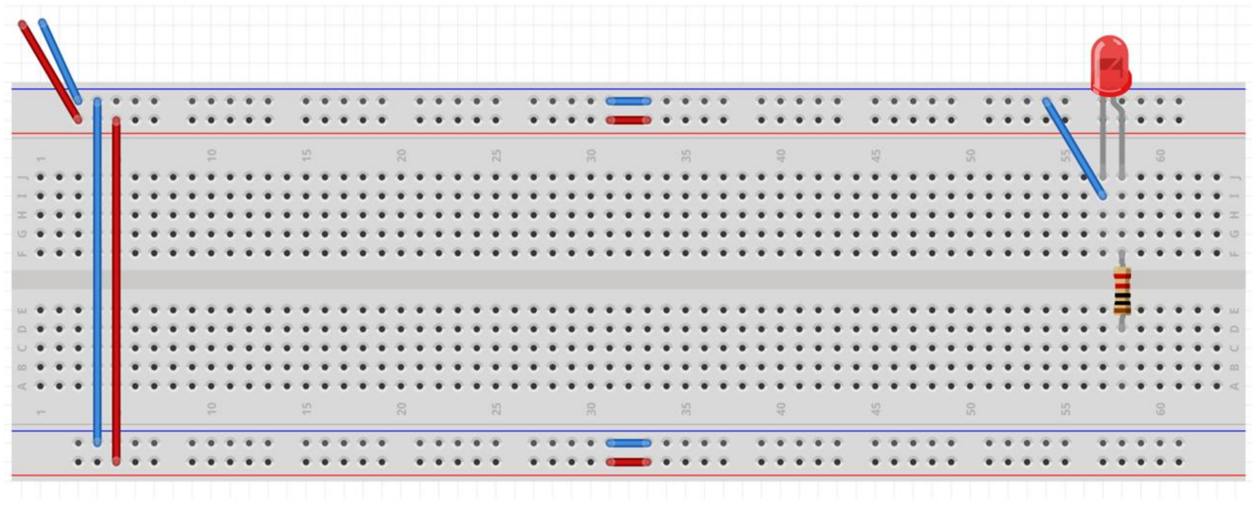
Setting up the output LED

These steps help you set up the LED that will be the "output" of your circuits. The negative side of the LED will always be connected to ground (low voltage). The positive side of the LED will have a high (True) or low voltage (False) depending on if we want to turn on or off the LED. LEDs require resistors to not burn out.

1. Find your 220 Ohm resistors. The color bands are red, red, black, black, brown (or reversed).
2. Plug a 220 Ohm resistor into the breadboard across the gap. Place it on the right side of the breadboard to leave room on the left for the chips.



3. Find a red LED. Plug the red LED with the long lead in the same column as the resistor. The "bent" leg in the diagram below is the long leg. It is the positive side.
4. Connect the short lead of the LED to the GND bus with a jumper wire.

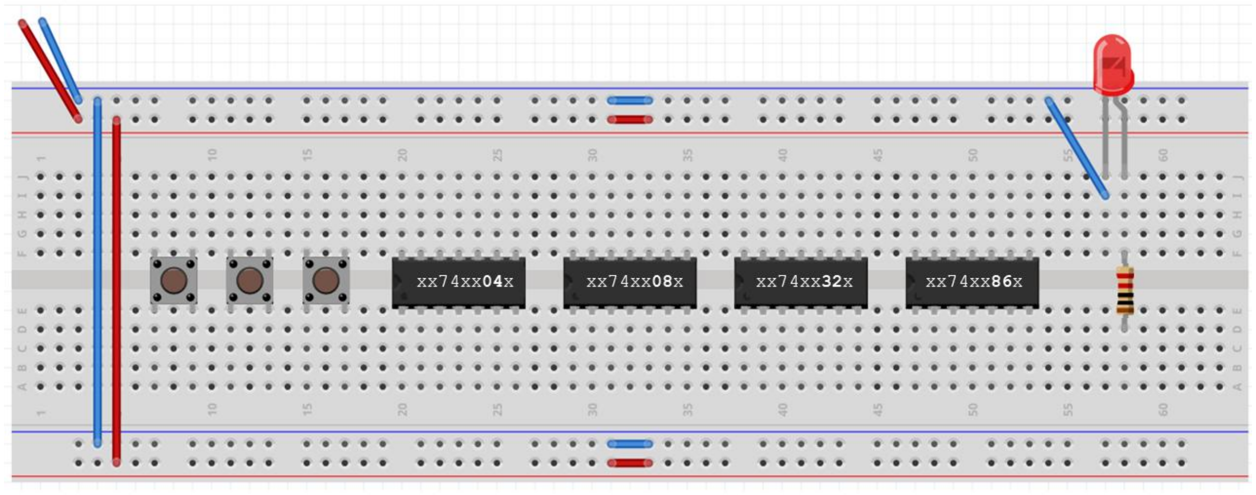


Throughout the lab, you will connect the output of the circuit you are testing to the resistor for this LED. Then, when power is on, you will observe the LED. If the LED is *on*, then it is receiving a high voltage, which maps to a logic *True* or a binary 1. If the LED is *off*, then it is receiving a low voltage, which maps to a logic *False* or a binary 0.

Part 1: Buttons as Inputs

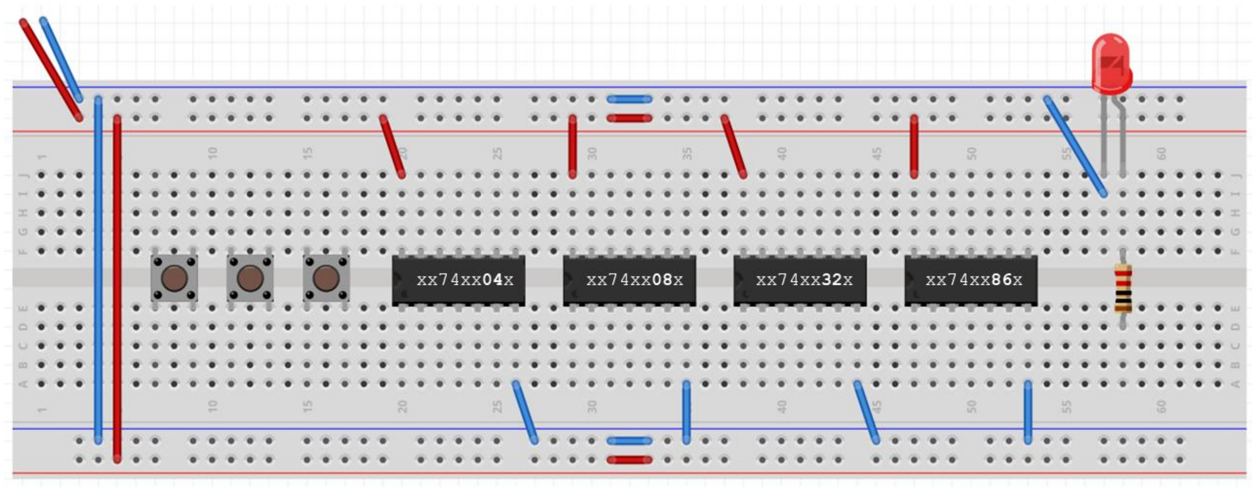
For this lab, instead of moving the input wires back and forth between the High Voltage Bus and the Low Voltage Bus, we will use push buttons. These are the same buttons we used in the Arduino Setup Lab.

You will need room on your breadboard for 3 push buttons, a NOT logic chip (74xx04), an AND logic chip (74xx08), an OR logic chip (74xx32), and an XOR logic chip (74xx86). Each of these components will straddle the middle gap. Put them all in now to make sure you have room for everything.

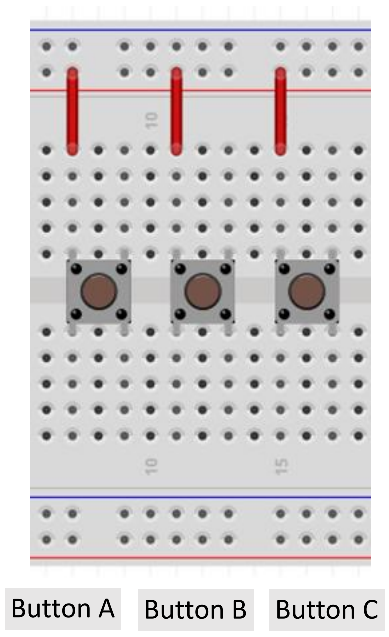


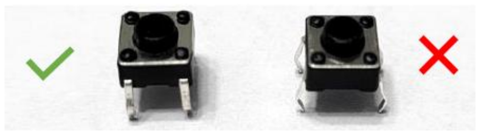
Note on chip removal: If you need to remove a logic chip from the board to reposition it somewhere else, be careful not to bend the pins. It should be pulled up as level as possible, i.e., don't lift one side higher than the other. The best way is to use a chip puller tool available in the labs on campus. If you don't have a chip puller, lift the chip off as straight as possible to avoid bending the pins. If you bend a pin, you should have a backup chip in your kit. Otherwise, ask your instructor for help.

Remember that each of the 74xx chips needs to be powered with a HIGH voltage on pin 14 (the top left pin) and LOW voltage on pin 7 (the bottom right pin). Add those wires now. (See image below.)



The push buttons each have 4 pins, one in each corner. The two pins on the left (top left and bottom left) are *already* connected to each other. Similarly, the two pins on the right (top right and bottom right) are *already* connected to each other. Pushing a button will connect its left pins to its right pins. Wire the buttons so that the left pins will be connected to a HIGH voltage (see image below).

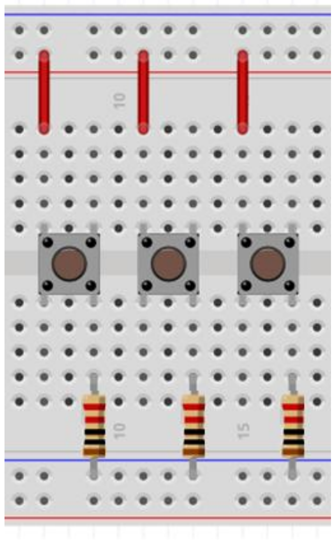




From now on, the instructions will refer to the three buttons as “Button A,” “Button B,” and “Button C.”

In later steps, the right pins will be connected to the inputs of the logic gates. That means that when you push a button, it will connect the input of the logic gate to a HIGH voltage. When the button is *not* pushed, the inputs need to be connected to a LOW voltage. (A disconnected wire doesn’t have a low voltage. It has what we call a *floating* voltage. If we want to pass a 0, we need to connect the wire to Ground.) However, we cannot connect the right side of the buttons directly to a LOW voltage, because then when the button is pushed, it will directly connect HIGH voltage on the left to LOW voltage on the right. We call that a *short circuit*, and it could fry your board.

To solve this problem, we will connect the right side of each button to Ground via a resistor to limit the current. When used this way, these are called *pull-down resistors.* Take three 220-ohm resistors (the same resistors as the LED uses) and use them to connect the right side of each button to Ground (LOW voltage). See the image below.

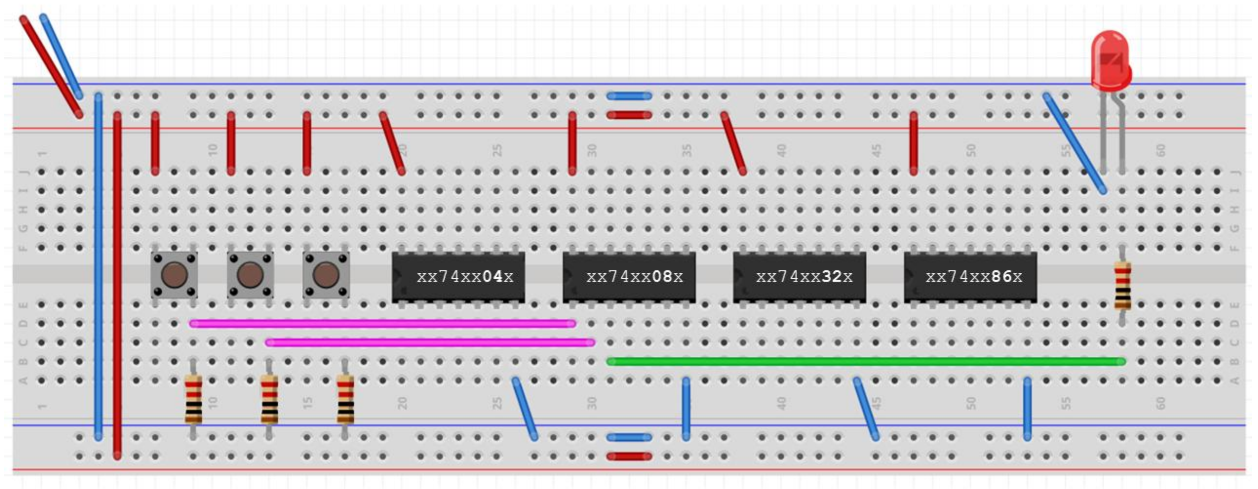


Now, when one of these buttons is pushed, its right pins will connect to the left pins and have a HIGH voltage, and when it is *not* pushed the right pins will have a LOW voltage, being connected to Ground through the resistor. The buttons can now be used as digital inputs, sending a HIGH or LOW signal when pushed or not pushed, respectively.

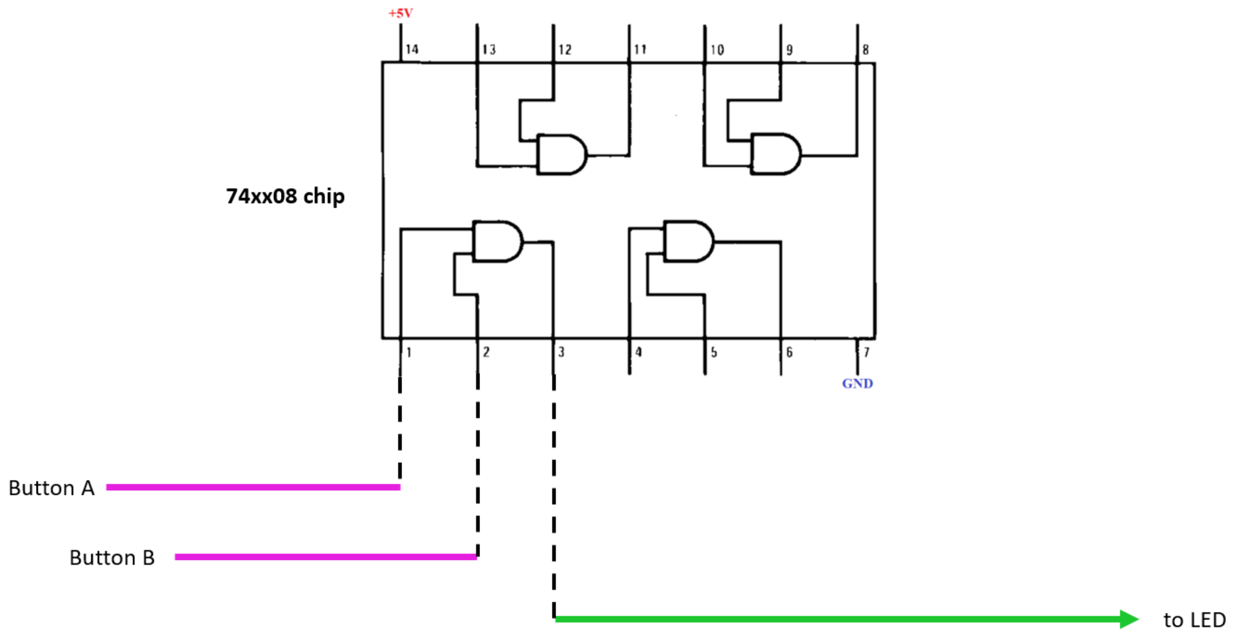
Test your buttons using one of the AND gates in the 74xx08 chip:

1. Connect the right pin of Button A to input pin 1 of the 74xx08 chip.
2. Connect the right pin of Button B to input pin 2 of the 74xx08 chip.
3. Connect output pin 3 of the 74xx08 chip to the LED's resistor.

Your circuit should look like the image below:



Review how the wiring of your circuit matches the schematic diagram below, referencing the AND gate you are using in the 74xx08 chip. (The dotted lines are connected vertically *inside* the breadboard because the wires you plug into the breadboard are in the same column as the respective pins off the chip.)



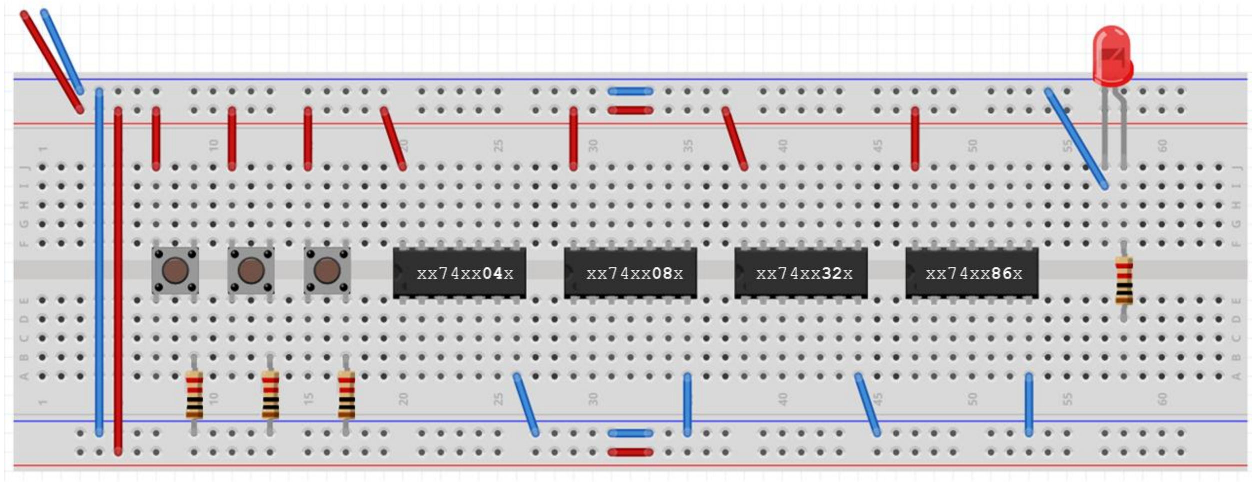
4. Test A=0, B=0.
    - After powering your board, you should observe that the LED is off when no buttons are pushed.
5. Test A=0, B=1.
    - Push Button B. The LED should still be off.
6. Test A=1, B=0.
    - Release Button B and push Button A. The LED should still be off.
7. Test A=1, B=1.
    - Push both Button A and Button B. The LED should be on.

If your circuit is not behaving correctly, try these troubleshooting tips:

- Double check the wiring.
- Double check your logic chip orientation.
- Double check your breadboard orientation.
- Double check your LED orientation.
- Double check that your logic chips are pushed in all the way and don’t have broken or bent pins.
- Double check that your logic chips don’t skip a row of holes above or below the gap.
- Make sure the buttons are all the way in.
- Try using one of the other AND gates in the 74xx08 chip.
- Try using a different 74xx08 chip.
- Make sure your breadboard is powered (see page 3).

If you were successful, continue. If not, ask your instructor or classmates for help.

8. **Take a photo**. You will submit this picture to “Part 1” in the I-Learn assignment.
9. Remove the three wires you just added during steps 1-3 above, leaving your circuit like the image below.



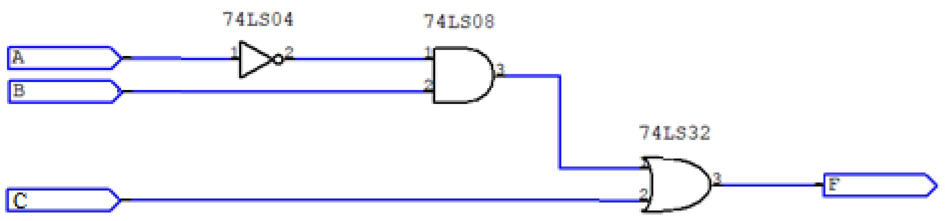
This setup will be the starting point for each new circuit that you build. Between each new circuit that you build in the remaining parts of the lab, you should return your circuit back to this setup. In other words, the wires that are now in your board can stay in the board until you are finished with the lab.

You have completed Part 1. Continue to the next page for Part 2.

Part 2: A Simple Logic Circuit

Next, you will combine logic gates together to create a logic circuit. Any Boolean function can be implemented with a logic circuit, and computer chips have many logic circuits inside to produce the desired Binary outputs for given Binary inputs. By combining many small and simple circuits, great things are brought to pass (see Alma 37:6).

For this part of the lab, you will build the simple logic circuit shown in this schematic:



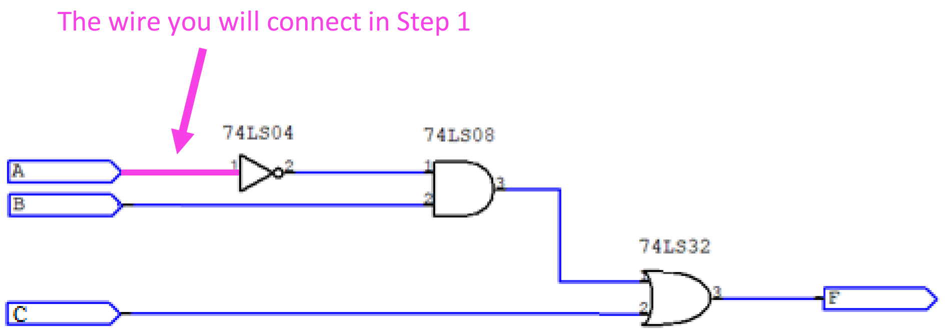
This logic circuit uses a NOT gate (in the 74xx04 chip), an AND gate (in the 74xx08 chip), and an OR gate (in the 74xx32 chip). This circuit implements the following Boolean equation: $\boldsymbol{F = \bar{A} \cdot B + C}$

The truth table for this circuit is shown below:

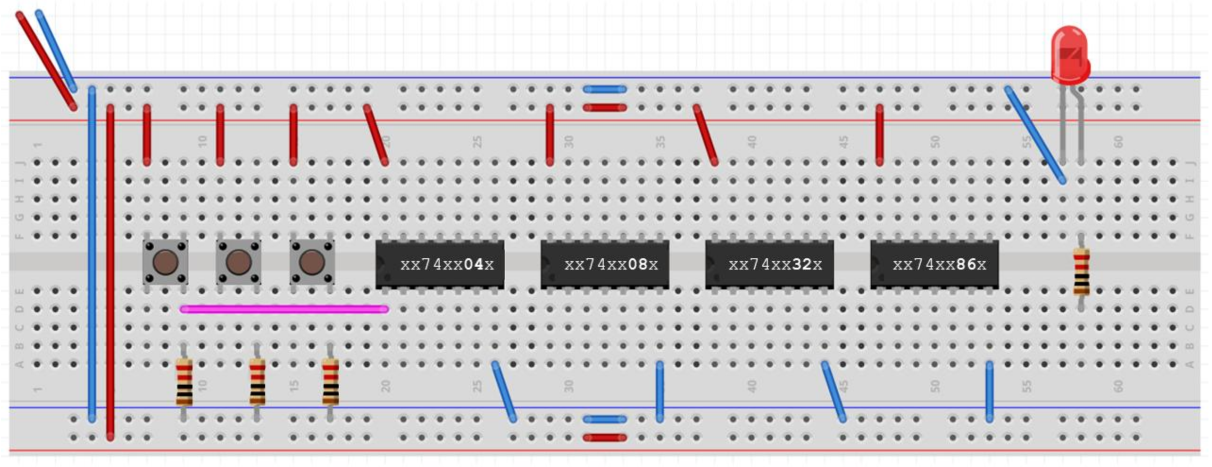
ABC	F
000	0
001	1
010	1
011	1
100	0
101	1
110	0
111	1

Notice that F is true whenever A is *False* <u>and</u> B is *True* (which is the $\bar{A} \cdot B$ term), <u>or</u> whenever C is true ($+C$).

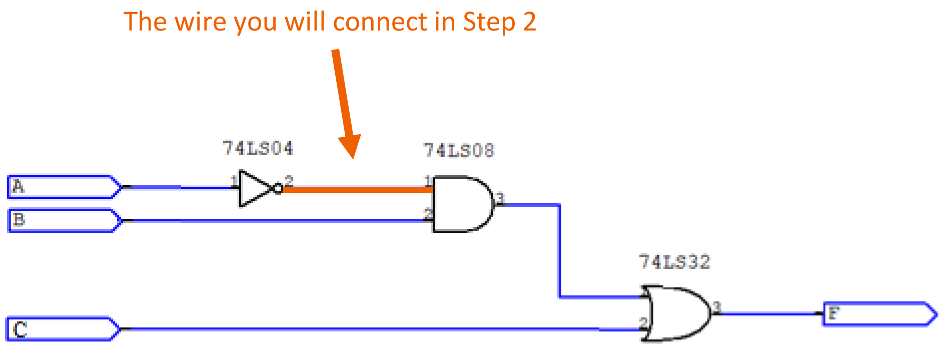
Button A will be input **A**. For Step 1, you will connect input A to one of the NOT gates in the 74xx04 chip.



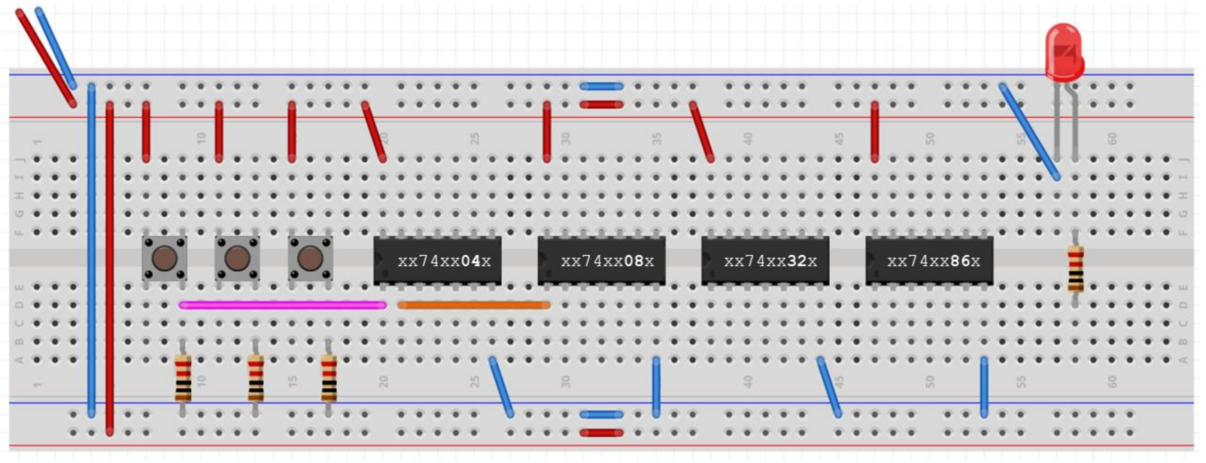
1. Connect the **right pin** of Button A to input **pin 1** of the 74xx04 NOT chip. That will send the **A** signal to the first NOT gate in the 74xx04 chip. Your circuit should look like the wiring diagram below.



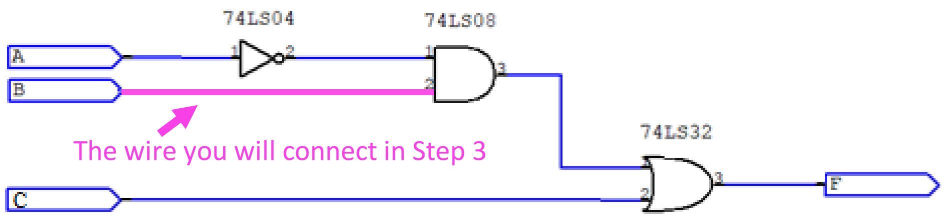
For Step 2, you will connect the output of that NOT gate as an input to one of AND gates in the 74xx08 chip.



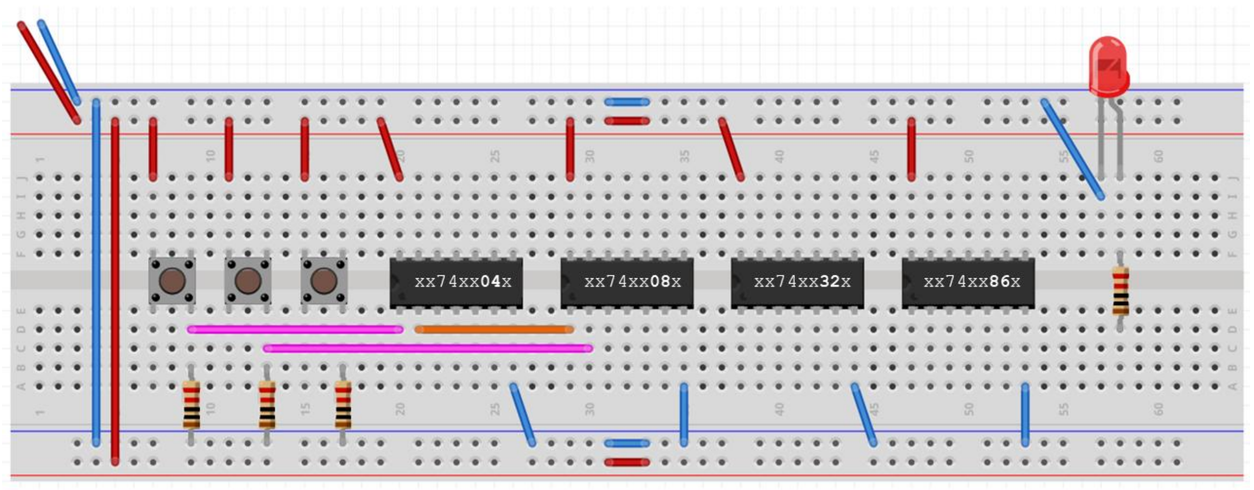
2. Connect output **pin 2** of the 74xx04 NOT chip to input **pin 1** of the 74xx08 AND chip. Your circuit should look like the wiring diagram below.



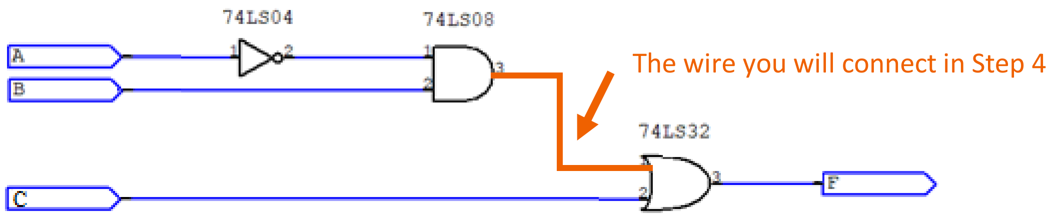
Button B will be input B. For Step 3, you will connect input B as the other input to that AND gate.



3. Connect the **right pin** of Button B to input **pin 2** of the 74xx08 AND chip. Your circuit should look like the wiring diagram below.

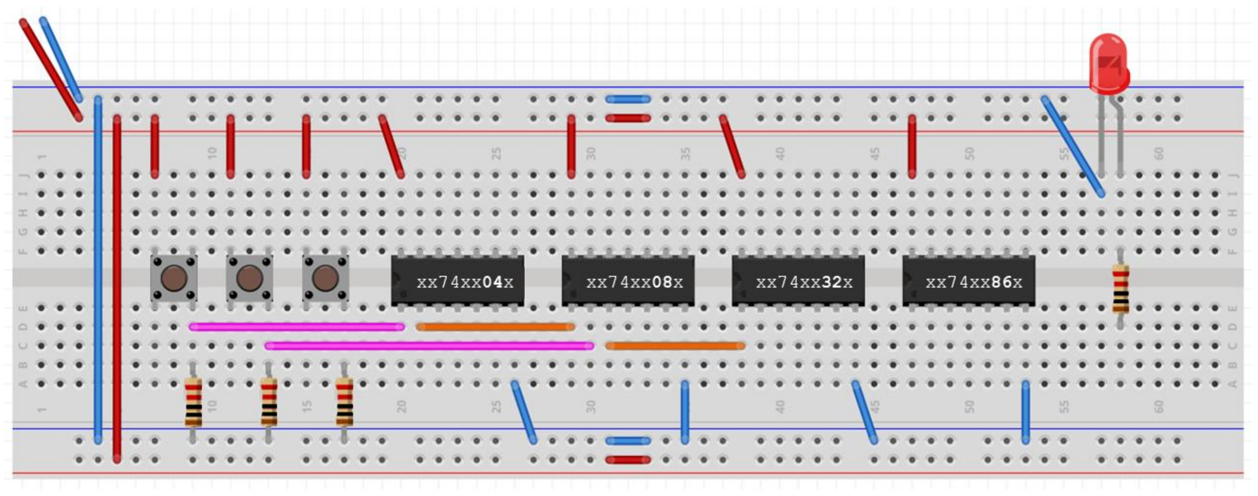


For Step 4, you will connect the output of that AND gate as an input to one of the OR gates in the 74xx32 chip.

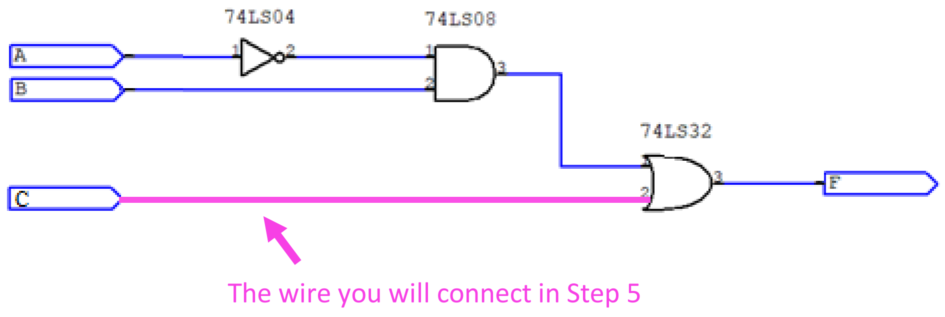


Note: Bends and curves in a wire do not affect how the wire is connected. What matters is the starting point and the ending point of the wire. In diagrams, we add bends to make the diagram more organized.

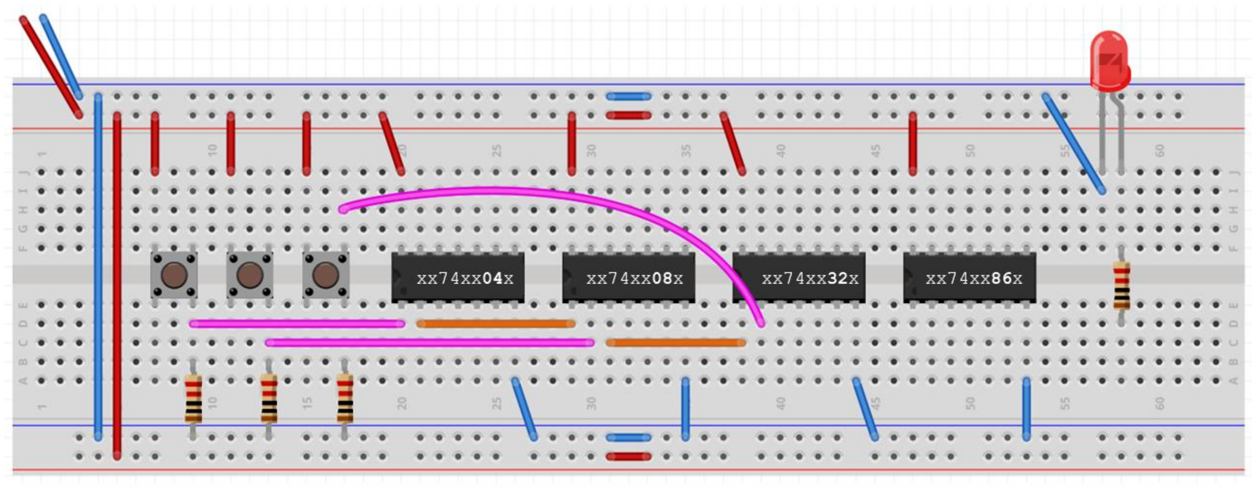
4. Connect output **pin 3** of the 74xx08 AND chip to input **pin 1** of the 74xx32 OR chip. Your circuit should now look like the following diagram.



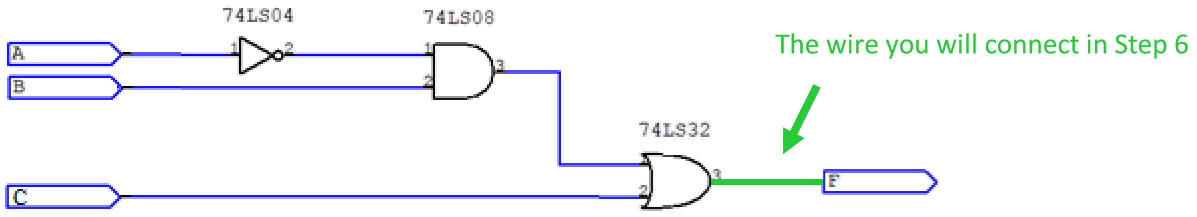
Button C will be input C. For step 5, you will connect input C to the other input of that OR gate.



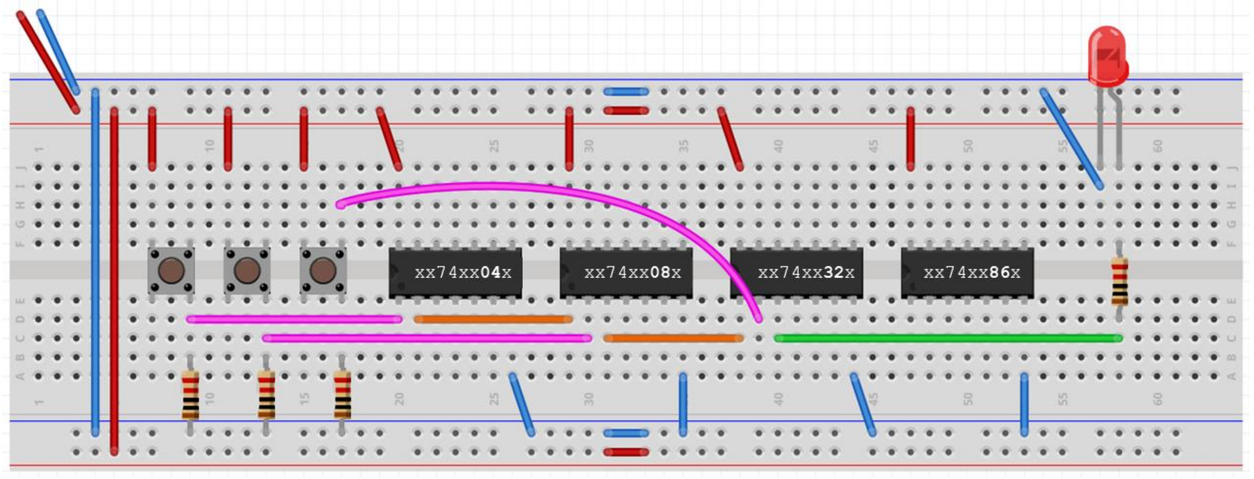
5. Connect the **right pin** of Button C to **pin 2** of the 74xx32 OR chip. Your circuit should now look like the wiring diagram below.



The final output signal **F** will go to the LED so we can observe when F is true (LED on) or false (LED off). For Step 6, you will connect the output of the OR gate to the LED's resistor.

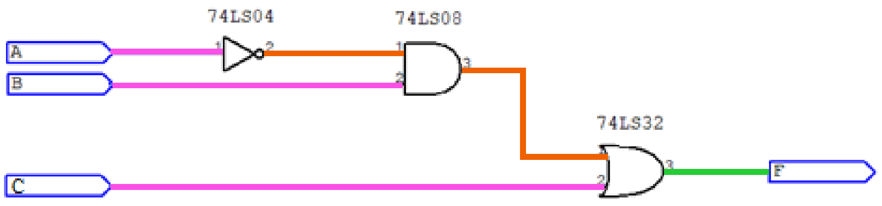


6. Connect p**in 3** of the 74xx32 to the bottom of the resistor going to the LED. Your circuit should now look like the following diagram.

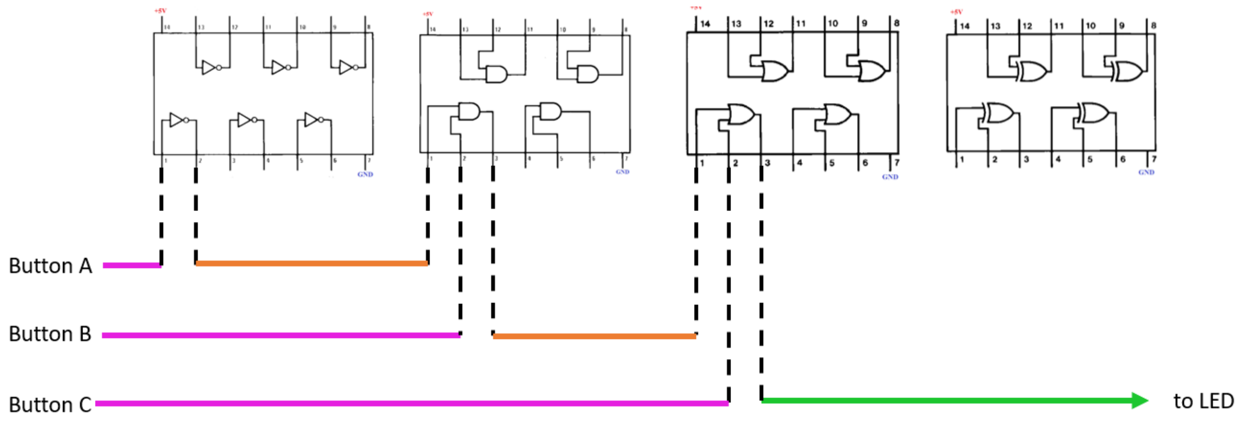


Input wires are shown in magenta. The output wire is shown in green. The gate-to-gate wires are shown in orange.

Before continuing, make sure you understand how the wiring diagram above matches the schematic below.



The following image may help you see the connections by referencing what is inside each chip.



Remember that your circuit is implementing this Boolean Equation: : $F = \overline{A} \cdot B + C$

That equation has this truth table:

ABC	F
000	0
001	1
010	1
011	1
100	0
101	1
110	0
111	1

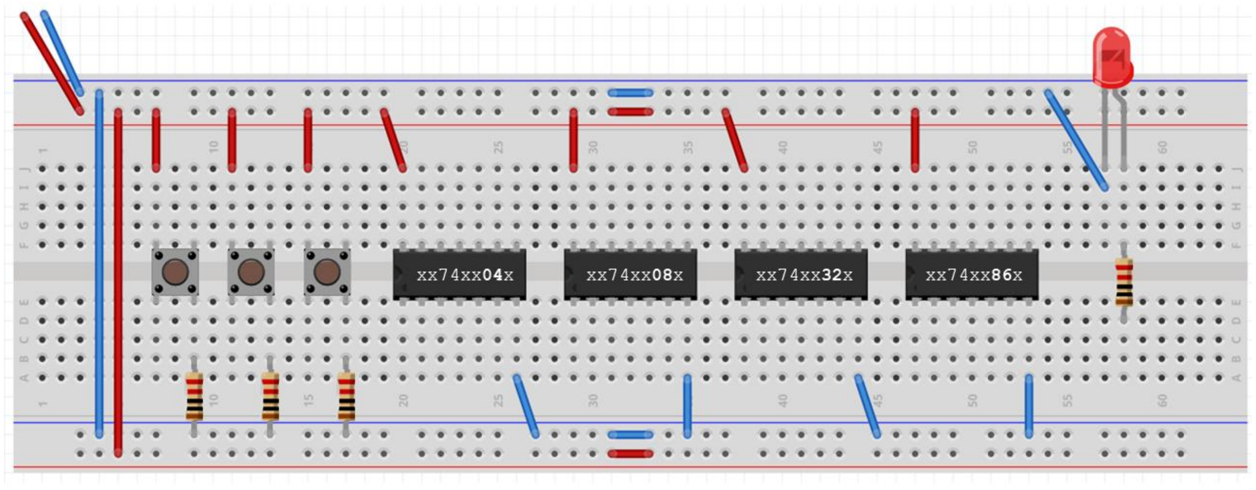
7. Test the circuit for proper operation by following these steps:
   a. Connect power to the breadboard by connecting your Arduino to your laptop.
   b. Test the first row of the truth table (A=0, B=0, C=0). The LED should be *off*.
   c. Test the next row of the truth table (A=0, B=0, C=1). The LED should be *on*.
   d. Test the third row of the truth table (A=0, B=1, C=0). The LED should be *on*.
   e. Continue testing each of the subsequent rows and comparing the LED with the truth table above.

If your tests were successful and match the truth table, record the values you observed in **truth table** in I-Learn. If they were not successful, doublecheck your circuit. Then, if your circuit is still not working correctly, ask your instructor for help.

Take a **photo** of your circuit to upload in I-Learn.

Disconnect power from the breadboard by disconnecting your Arduino from your laptop.

Remove the wires you added during Part 2, but leave the following wires:



You have completed Part 2. Continue to the next page for Part 3.

Part 3: A Circuit You Design

For this final part of the lab, design your own 3-input logic circuit, different than the circuit in Part 2, that uses multiple logic gates.*

Build the circuit on your breadboard. Use Buttons A, B, and C as the inputs and the LED as the output.

Test: After wiring your circuit, power the board and test all 8 possible input combinations. Make sure the values you observe match what you expect given your equation.

Record:

- Record the values you observed in the **truth table** in I-Learn.
- Determine the Boolean equation that exactly matches your circuit. **Enter the equation** in I-Learn.
- Take a **photo** of your circuit to upload to I-Learn. Power off the board.

*If the circuit you designed needs gates that you don't have, such as a 3-input AND gate or a NAND gate, see the schematics in Optional Parts 4 and 5 for help.

Important: Even if you choose not to do the optional activities below, you should still read through the descriptions and review the schematics in each optional activity to learn those concepts.

Optional Activities

You are now finished with the lab. If you would like to stretch yourself, choose one or more of the following *optional* activities to complete.

- Part 4: Creating Three-Input Gates from Two-Input Gates (pages 16-17)
- Part 5: De Morgan Circuits (pages 18-20)
- Part 6: A Multiplexor Circuit (page 21)

When you are finished, remove all wires, buttons, resistors, and the LED. You may leave the chips in the board.

Optional Part 4: Three-Input Gates from Two-Input Gates

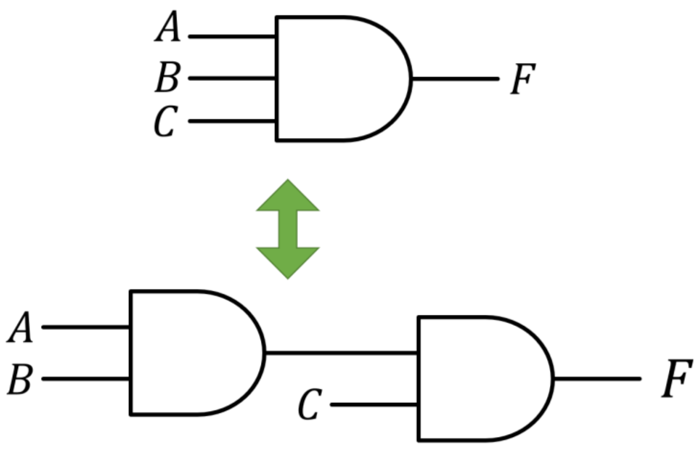
There exist gates that are 3-input, 4-input, 5-input, etc., but your kit only has 2-input gates. Review this appendix in case you need a 3-input gate for your circuit. You can create a 3-input gate by combining two 2-input gates.

Part 4(A): A Three-Input AND Circuit

You can construct a circuit that performs a 3-input AND operation by combining two 2-input AND gates.

Equation: $\boldsymbol{F = A \cdot B \cdot C = (A \cdot B) \cdot C}$ ←This is called the *associative property*.

Circuit Schematic: (Your circuit will match the bottom, which will perform the same operation as the top.)

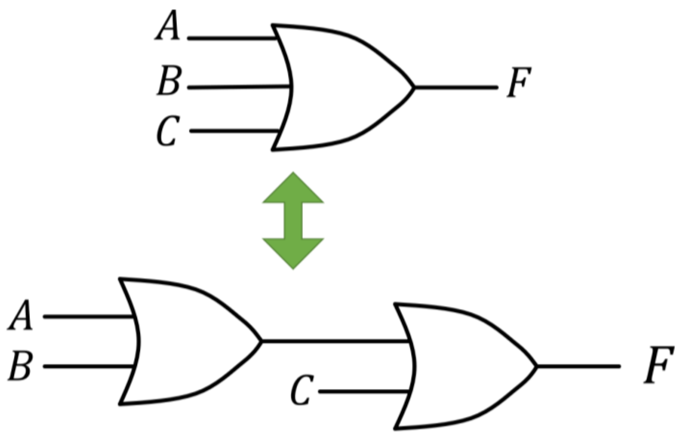


Part 4(B): A Three-Input OR Circuit

You can construct a circuit that performs a 3-input OR operation by combining two 2-input OR gates.

Equation: $\boldsymbol{F = A + B + C = (A + B) + C}$

Circuit Schematic: (Your circuit will match the bottom, which will perform the same operation as the top.)



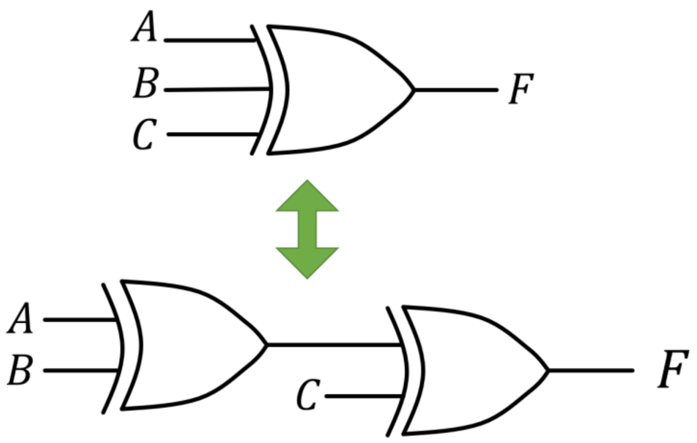
Part 4(C): A Three-Input XOR Circuit

You can construct a circuit that performs a 3-input XOR operation by combining two 2-input XOR gates.

Note on XOR gates with more than 2 inputs: When an XOR gate has only two inputs, the output is high if and only if ***exactly* one** of the inputs is high. However, an XOR gate can have more than two inputs. The general rule for XOR gates is that the output is high if and only if there are **an *odd* number of inputs** that are high. So, for a 3-input XOR gate, the output will be high if there is 1 high input or if there are 3 high inputs. If there are no high inputs or 2 high inputs, the output will be low.

Equation: $\boldsymbol{F = A \oplus B \oplus C = (A \oplus B) \oplus C}$

Circuit Schematic: (Your circuit will match the bottom, which will perform the same operation as the top.)



Optional Part 5: De Morgan Circuits

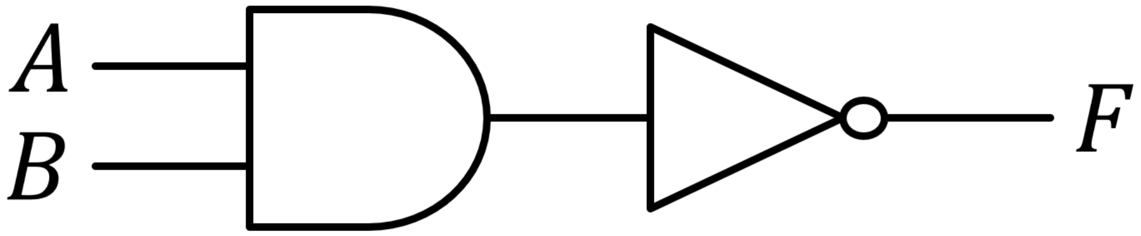
This part is optional if you'd like to, you may wire up 6 simple circuits – 5(A) through 5(F) – to demonstrate a property called *De Morgan's* rule. As you test these circuits, you will observe the difference of inverting the inputs to a gate versus inverting the output of a gate.

Part 5(A): NAND

Build a circuit that performs a NAND operation using an AND gate followed by a NOT gate. Use Button A as Input A and Button B as Input B. Use any of the AND gates in the 74xx08 chip. Use any of the NOT gates in the 74xx04 chip.

Equation: $\boldsymbol{F = \overline{A \cdot B} = (AB)'}$

Circuit Schematic:



Test: After wiring your circuit, power the board and test the following:

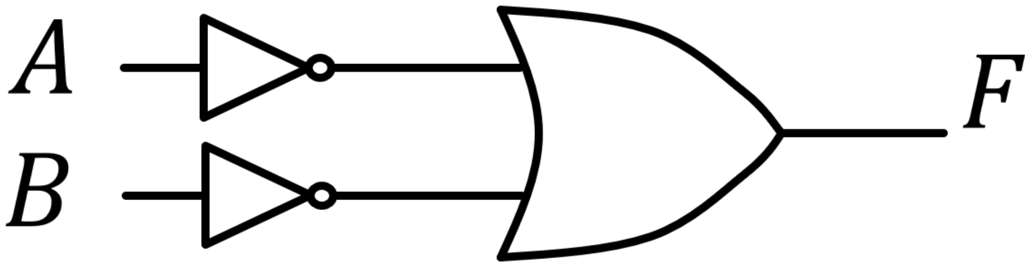
- A=0, B=0 (neither button pushed) **- output should be *True***
- A=0, B=1 (only button B pushed) **- output should be *True***
- A=1, B=0 (only button A pushed) **- output should be *True***
- A=1, B=1 (both buttons pushed) **- output should be *False***

Part 5(B): Another way to do NAND

Build a circuit that performs a NAND operation using two NOT gates inverting the inputs into an OR gate.

Equation: $\boldsymbol{F = \overline{A} + \overline{B} = A' + B'}$

Circuit Schematic:



Test: After wiring your circuit, power the board and test all 4 input combinations using buttons A and B.

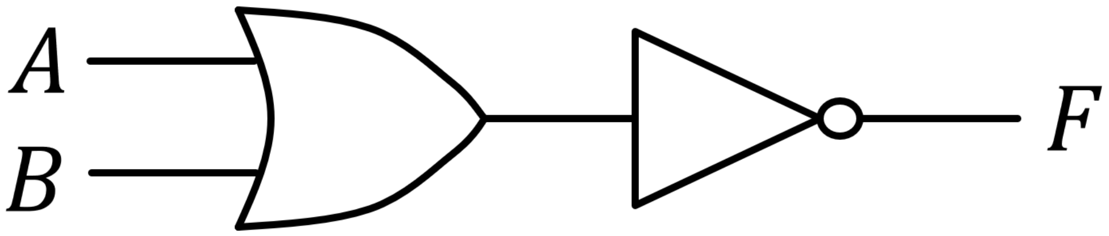
- A=0, B=0 (neither button pushed) **- output should be *True***
- A=0, B=1 (only button B pushed) **- output should be *True***
- A=1, B=0 (only button A pushed) **- output should be *True***
- A=1, B=1 (both buttons pushed) **- output should be *False***

Part 5(C): NOR

Build a circuit that performs a NOR operation using an OR gate followed by a NOT gate. Use Button A as Input A and Button B as Input B. Use any of the OR gates in the 74xx32 chip. Use any of the NOT gates in the 74xx04 chip.

Equation: $\boldsymbol{F = \overline{A + B} = (A + B)'}$

Circuit Schematic:



Test: Connect the F wire to the LED's resistor. After wiring your circuit, power the board and test the following:

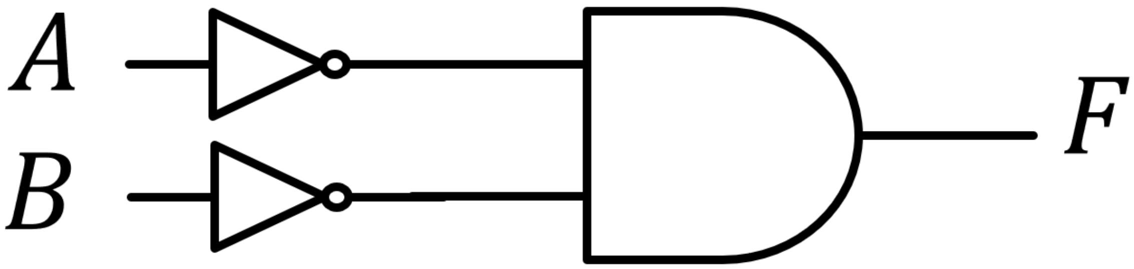
- A=0, B=0 (neither button pushed) **- output should be *True***
- A=0, B=1 (only button B pushed) **- output should be *False***
- A=1, B=0 (only button A pushed) **- output should be *False***
- A=1, B=1 (both buttons pushed) **- output should be *False***

Part 5(D): Another way to do NOR

Build a circuit that performs a NOR operation using two NOT gates inverting the inputs into an AND gate. Use any two of the NOT gates in the 74xx04 chip. Use any of the AND gates in the 74xx08 chip.

Equation: $\boldsymbol{F = \overline{A} \cdot \overline{B} = A'B'}$

Circuit Schematic:



Test: Connect the F wire to the LED's resistor. After wiring your circuit, power the board and test the following:

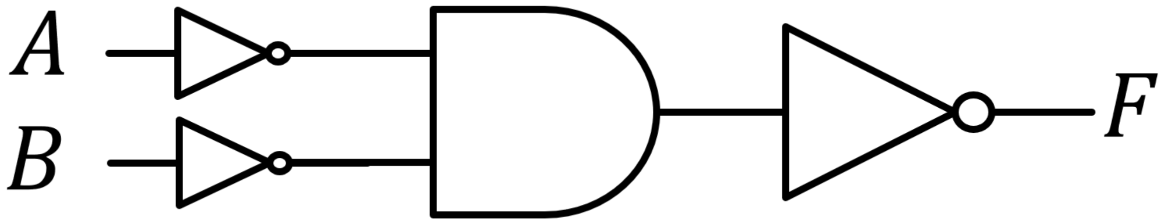
- A=0, B=0 (neither button pushed) **- output should be *True***
- A=0, B=1 (only button B pushed) **- output should be *False***
- A=1, B=0 (only button A pushed) **- output should be *False***
- A=1, B=1 (both buttons pushed) **- output should be *False***

Part 5(E): Another way to do OR

Build a circuit that performs an OR operation using two NOT gates inverting the inputs into an AND gate followed by another NOT gate inverting the output.

Equation: $\boldsymbol{F = \overline{\overline{A} \cdot \overline{B}} = (A'B')'}$

Circuit Schematic:



Test: After wiring your circuit, power the board and test all 4 input combinations using buttons A and B.

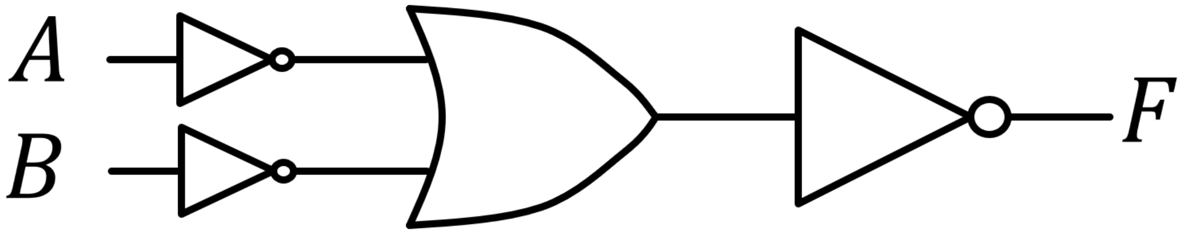
- A=0, B=0 (neither button pushed) **- output should be *False***
- A=0, B=1 (only button B pushed) **- output should be *True***
- A=1, B=0 (only button A pushed) **- output should be *True***
- A=1, B=1 (both buttons pushed) **- output should be *True***

Part 5(F): Another way to do AND

Build a circuit that performs an AND operation using two NOT gates inverting the inputs into an OR gate followed by another NOT gate inverting the output.

Equation: $\boldsymbol{F = \overline{\overline{A} + \overline{B}} = (A' + B')'}$

Circuit Schematic:



Test: After wiring your circuit, power the board and test all 4 input combinations using buttons A and B.

- A=0, B=0 (neither button pushed) **- output should be *False***
- A=0, B=1 (only button B pushed) **- output should be *False***
- A=1, B=0 (only button A pushed) **- output should be *False***
- A=1, B=1 (both buttons pushed) **- output should be *True***

De Morgan's Rule: You should have seen that inverting the inputs to a gate is *not* the same thing as inverting the output of that same gate. For example, Part 5(A) does *not* have the same truth table as Part 5(D). Instead, inverting the output of an AND gate is the same as inverting the inputs of an OR gate – Part 5(A) has the same truth table as Part 5(B). De Morgan's Rule may be even more apparent by seeing that inverting both the inputs *and* the output of an AND gate makes it the same as an OR gate (Part 5E) and inverting the both the inputs and the outputs of an OR gate makes it the same as an AND gate (Part 5F).

Optional Part 6: A Multiplexor Circuit

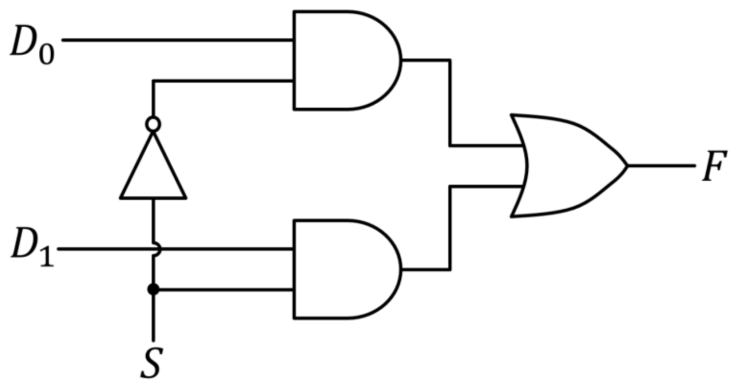
You will now optionally connect a circuit that is very common in computer chips called a *multiplexor*. A multiplexor is used to select one of multiple input signals that it will allow to pass through to the output signal. Your circuit will have two Data inputs (D_0 and D_1) and one Selector input (S). If the Selector is 0, then the circuit will pass the value in D_0 to the output F. If the Selector is 1, then the circuit will pass the value in D_1 to the output F.

Equation: $\boldsymbol{F = D_0\overline{S} + D_1 S}$

In other words,

- If S equals 0, then $F = D_0$
- If S equals 1, then $F = D_1$

Circuit Schematic:



Use Button A for $\boldsymbol{D_0}$. Use Button B for $\boldsymbol{D_1}$. Use Button C for $\boldsymbol{S}$. Send the output $\boldsymbol{F}$ to the LED.

Test: After wiring your circuit, power the board and test all 8 possible input combinations using the 3 buttons. You should see the same results as in this truth table:

D_0	D_1	S	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Another way of writing this same truth table which better shows the purpose of this circuit is like this:

S	F
0	D_0
1	D_1

Now, by controlling S, we can choose if F equals D_0 or if it equals D_1.