**Score: \_\_\_\_\_**

**Lab03 – Programmable Circuits**

COMP256 – Computing Abstractions

Dickinson College

Spring 2023

Prof. Grant Braught

**Name(s):**

**Introduction:**

In the first lab you built build one of the fundamental units of computation, a logical NOT gate, using transistors. In the second lab you used designed a circuit that could perform a useful computation - adding three 1-bit numbers. You then used integrated circuit chips to connect logic gates to build that circuit.

In this lab, you will build on what you know to create a circuit that is “programmable.” This circuit will use a pattern of 1’s and 0’s to tell it what function to perform on it’s inputs. While this may not quite seem like “programming,” if you think of the pattern of 1’s and 0’s as an instruction (e.g. 01 meaning compute OR, or 11 meaning compute a sum) then it may seem more plausible. After all, any of the instructions we write in a language like Java (e.g. z = x + y) must be ultimately translated into instructions that are represented using 1’s and 0’s that tell the machine what to do. We’ll be learning more about that soon. But for now, you’ll get a little insight into how a pattern of 1’s and 0’s can act as an instruction to a machine.

**A Programmable 1-bit Machine:**

The programmable machine we will be building performs computations on two 1-bit inputs, we will call them A and B. The machine will be able to perform four different computations on those inputs: NOT A, A OR B, A AND B, and A PLUS B. Which operation this simple computer performs will be controlled by a 2-bit instruction. We’ll call the bits of the instruction S1 and S0 ,so that we can easily refer to them.

The bits of the instruction control the operation that is performed (i.e. the output that is generated) by the machine according to the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Instruction** | | **Operation** |  |
|  | S1 | S0 | **Performed** |  |
|  | 0 | 0 | NOT A |  |
|  | 0 | 1 | A OR B |  |
|  | 1 | 0 | A AND B |  |
|  | 1 | 1 | A PLUS B |  |

For example, the instruction 10 will cause the machine to perform the operation A AND B. So, if the inputs to the machine are A=0, B=1 then the Instruction 10 (i.e. S1=1 and S0=0) would generate a result of 0 because 0 AND 1 is 0. A few additional notes on these operations:

* When NOT A is computed the input B is ignored.
* When A PLUS B is computed we will only care about the sum, any carry that would occur is ignored (but we could add to our circuit to compute it too!)

🔑 1. Fill in the each row of the Result column in the following table with a 0 or a 1 to show the result that would be generated if the indicated instruction is performed on the given inputs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Instruction** | | **Inputs** | |  |  |
|  | S1 | S0 | A | B | **Result** |  |
|  | 0 | 0 | 1 | 0 |  |  |
|  | 0 | 1 | 1 | 0 |  |  |
|  | 1 | 1 | 1 | 0 |  |  |
|  | 1 | 0 | 1 | 1 |  |  |
|  | 1 | 1 | 1 | 1 |  |  |
|  |  |  |  |  |  |  |

**Making Choices:**

One of the fundamental operations in programmable machines is that of making a choice. For example, we will have to choose the operation to perform (NOT, OR, AND or PLUS) based on the bits in the instruction. Thus, to create a programmable machine we will need to build a circuit that can make a choice. The next several questions will lay the foundations for making choices in circuits and then you will construct a circuit that can make a choice.

2. Consider the following logic circuit:



a. Complete the truth table for this circuit below. Include the intermediate variables X and Y and the output R.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **C** | **P** | **Q** | **X** | **Y** | **R** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

🔑 b. You should notice that when C is 0 the output (R) matches whatever P is and conversely that when C is 1 the output (R) matches whatever Q is.

*No answer is required here, but your circuit must function as described above. If it does not, revisit part a and resolve any errors.* ***If you are unsure if your circuit or table is correct, have your instructor check it at this point.***

🔑 c. To make it a little clearer that this circuit is making a choice based on an instruction, let’s describe the behavior of this circuit as an abstraction. To do so, fill in the blank in each of the sentences below with either P or Q:

If the instruction C = 0 then the circuit chooses as its output R.

If the instruction C = 1 then the circuit chooses as its output R.

Based on the statements you just filled, you should see that the value of the instruction (C) is causing this circuit to make a choice between outputting either P or Q. So, in some very real sense, **this circuit is programmable!**

It is worth mention that the circuit above is very common. So common that it has its own name. It is called a *multiplexer* and there are integrated circuit chips (e.g. a 74HC157 chip) that abstract away the details of the logic gates that implement it. Unfortunately, TINKERCAD does not provide this chip, so we will have to build our multiplexers from gates.

**Choosing Operations:**

We can now build on the above circuit to create a larger circuit that makes a choice between two different computations. We will start with just the first two operations that our computer can perform (NOT A, and A OR B).

🔑 3. Consider the schematic diagram shown below. It builds directly on the circuit from question #2, which is highlighted in the yellow box. This diagram just changes the labels a bit. C is changed to be S0, one of the bits of our instruction. R is changed to be Z0, the result of the computation indicated by the instruction S0.



Now we can build on your answers from question #2c to make more clear that this circuit is doing a computation. Complete the following sentences using the input variables (A, B) and the operations (NOT, OR).

The instruction S0 = 0 causes the circuit to compute as its output Z0.

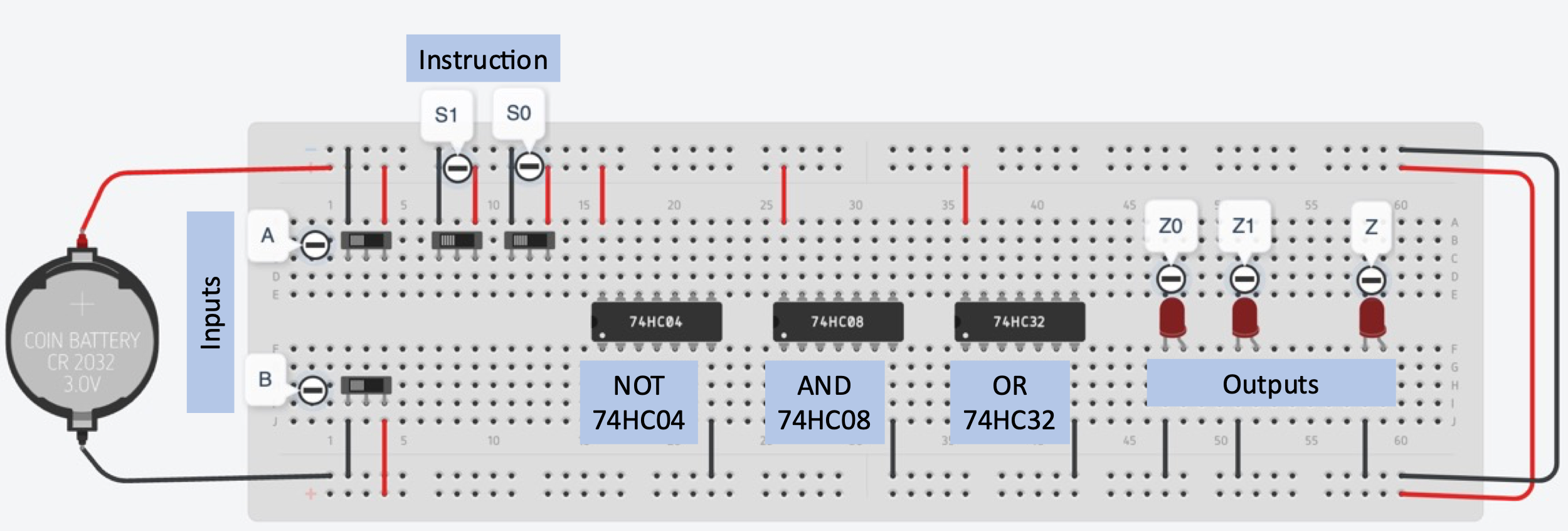
The instruction S0 = 1 causes the circuit to compute as its output Z0.

4. Create a new circuit in TINKERCAD named Lab03-NotOr.

*You’ll give a link to your TINKERCAD circuit below when it is complete.*

5. Setup a regular size (not small or mini) breadboard as shown below. Note: I added the blue labels to this image for clarity. You do not need to create those labels. You should however create the white labels in TINKERCAD.

*You’ll give a link to your TINKERCAD circuit below when it is complete.*



🔑 6. Using the components on this breadboard, construct the circuit illustrated by the schematic in question #3.

* **References for the pin-out diagrams for all of the integrated circuit chips that you need can be found at the end of this lab.**
* Don’t forget to make the Vcc and ground connections to each chip.
* You will use the slide switches A, B as your inputs. Use labels to indicate which switch is A and which switch is B.
* You will use the slide switch S0 as the S0 bit of the instruction. You will not need to use the slide switch S1 until later in the lab.
* Connect the output of the circuit (Z0) to the “bent leg” of the LED. You will not need to use the other outputs Z1 and Z until later in the lab.

Give a link to your working TINKERCAD circuit here.

7. Using your circuit complete the following table:

* In the LED column, indicate if the LED is on or off for each input combination.
* In the Z0 column give the logic value (0 or 1) at the circuit output. You can measure it with a voltmeter or recall that the LED is on if the value at its “bent leg” is 1 and off if that value is 0.
* In the Function column, give a logic expression that expresses the function that has been computed. For example Z0 = NOT A or Z0 = A OR B, etc…. Note that the same function will be listed multiple times.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **S0** | **A** | **B** | **LED** | **Z0** | **Function** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

🔑 8. Compare the results you have in the above table with your answer to question #3 above. If the results are not consistent, revisit questions #3-7 and resolve any issues.

*No answer is required for this question, but your circuit must function correctly before going on.* ***If you are unsure, ask your instructor at this point.***

**The Other Two Operations:**

Notice that the above circuit handles the first two operations of our programmable 1-bit machine (NOT A and A OR B). You will now build a circuit that handles the second two operations (A AND B and A PLUS B).

9. To handle the second two operations, we will need to compute A AND B, which is pretty easy using an AND gate. But we also need to compute A PLUS B. It turns out that this is pretty easy as well.

a. Complete the following truth table with the value of A PLUS B. Note: We are ignoring the carry that might result for this lab so (1 PLUS 1 is just 0).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | A | B | A PLUS B |  |
|  | 0 | 0 |  |  |
|  | 0 | 1 |  |  |
|  | 1 | 0 |  |  |
|  | 1 | 1 |  |  |
|  |  |  |  |  |

🔑 b. Which logic gate computes A PLUS B as shown in the table above? Hint: You can go back to your earlier homework and check the on-line reference or the slide from class that shows all of the logic gates and their truth tables.

10. Now we can create a circuit that will compute the second two functions.

a. Draw a schematic diagram for a circuit that will use the instruction bit (S0) so that the circuit chooses A AND B when S0 = 0 and chooses A PLUS B when S0 = 1. Label the output of this circuit Z1 to differentiate it from the earlier output Z0. Hint: Think about how you could adapt the schematic in question #3 to create a different circuit that performs this computation!

🔑 b. Complete the following sentences using the input variables (A, B) and the operations (AND, PLUS).

The instruction S0 = 0 causses this circuit to compute as its output Z1.

The instruction S0 = 1 causes this circuit to compute as its output Z1.

11. Create a copy of your Lab03-NotOr circuit in TINKERCAD and rename it Lab03-AndPlus.

*You’ll give a link to your TINKERCAD circuit below when it is complete.*

🔑 12. Add a second full-size breadboard to your Lab03-AndPlus circuit, leaving the original one unchanged.

On this new breadboard, implement the circuit you drew in question #10.

* **Do not add a second battery. Connect the power and ground rails of your new breadboard to the same battery as the first.**
* Do not add any new slide switches:
  + Use the A and B slide switches from your original bread board as the inputs for this circuit. Just run a wire from your original breadboard to the new one.
  + Use the S0 slide switch from your original bread board as the instruction to for circuit. Just run a wire from your original breadboard to the new one.
* Use the LED from your original bread board to show the output Z1, similar to how the LED in your original circuit showed the output Z0.

Give a link to your working TINKERCAD circuit here.

13. Using your new circuit complete the following table, similar to the one earlier.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  | **S0** | **A** | **B** | **LED** | **Z1** | **Function** |  |
|  | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |

🔑 14. Compare the results you have in the above table with your answer to question #10b above. If the results are not consistent, revisit questions #9-13 and resolve any issues.

*No answer is required for this question, but your circuit must function correctly before going on.* ***If you are unsure, ask your instructor at this point.***

**Putting it all Together:**

Now let’s look at the table of instructions from the beginning of the lab again in a little different way:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Instruction** | | **Operation** | **Final** |  |
|  | S1 | S0 | **Performed** | **Output (Z)** |  |
|  | 0 | 0 | Z0 = NOT A | Z0 |  |
|  | 0 | 1 | Z0 = A OR B |  |
|  | 1 | 0 | Z1 = A AND B | Z1 |  |
|  | 1 | 1 | Z1 = A PLUS B |  |

Here we can see that the instruction bit S0 decides if the output Z0 will be NOT A (when S0=0) or A OR B (when S0 =1). But notice that S0 also decides if Z1 will be either A AND B (when S0=0) or A PLUS B (S0=1). This behavior corresponds exactly to the two circuits that you have constructed above. **If you don’t quite see how, ask your instructor.**

Now notice that the instruction bit S1 chooses the final output (Z). When the instruction bit S1=0, the final output (Z) will be whatever Z0 is (either NOT A or A OR B depending on S0). Similarly, when the instruction bit S1=1, the final output (Z) will be whatever Z1 is (either A AND B or A PLUS B depending on S0). So collectively, the instruction bits S1 and S0 decide which of the four possible computations (NOT A, A OR B, A AND B, A PLUS B) will become the final output (Z). **If you don’t quite see how, ask your instructor.**

Your two earlier circuits computed Z0 and Z1. So, all that is needed now is a circuit that uses the instruction bit S1 to choose either Z0 or Z1. And… we know from question #3 that we can build a circuit that uses an instruction bit to choose between two different inputs!

15. Create a copy of your Lab03-AndPlus circuit in TINKERCAD and rename it

Lab03-1BitComputer.

*You’ll give a link to your TINKERCAD circuit below when it is complete.*

🏆 16. Without changing the circuits you have already built, add another breadboard (the small is probably fine this time) to your Lab03-1BitComputer circuit.

On this breadboard, construct a circuit that generates the final result of the computation (Z). This circuit will just need to use instruction bit S1 to choose between the output of your first circuit (i.e. Z = Z0 when S1=0) or the output of your second circuit (i.e. Z=Z1 when S1=1) using the value of S1.

* Connect the rails on your new breadboard to the same battery as the others. Do not add a new battery.
* Use the LED on your original bread board to show the final output (Z).
* Use the S1 slide switch on your original bread board for the instruction bit S1.
* Use the A, B and S0 slide switches from your original breadboard as inputs to this circuit.

Give a link to your working TINKERCAD circuit here.

17. Experiment with your circuit to complete the following truth table where Z is the final output:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  | **S1** | **S0** | **A** | **B** | **LED** | **Z** | **Function** |  |
|  | 0 | 0 | 0 | 0 |  |  |  |  |
|  | 0 | 0 | 0 | 1 |  |  |  |  |
|  | 0 | 0 | 1 | 0 |  |  |  |  |
|  | 0 | 0 | 1 | 1 |  |  |  |  |
|  | 0 | 1 | 0 | 0 |  |  |  |  |
|  | 0 | 1 | 0 | 1 |  |  |  |  |
|  | 0 | 1 | 1 | 0 |  |  |  |  |
|  | 0 | 1 | 1 | 1 |  |  |  |  |
|  | 1 | 0 | 0 | 0 |  |  |  |  |
|  | 1 | 0 | 0 | 1 |  |  |  |  |
|  | 1 | 0 | 1 | 0 |  |  |  |  |
|  | 1 | 0 | 1 | 1 |  |  |  |  |
|  | 1 | 1 | 0 | 0 |  |  |  |  |
|  | 1 | 1 | 0 | 1 |  |  |  |  |
|  | 1 | 1 | 1 | 0 |  |  |  |  |
|  | 1 | 1 | 1 | 1 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

🔑 18. Double check each of the rows of the above truth table to confirm that your circuit is working correctly. If anything is amiss, revisit questions #15-17 and resolve any issues.

*No answer is required for this question, but your circuit must function correctly before going on.* ***If you are unsure, ask your instructor at this point.***

**Some Final Thoughts:**

So, I don’t know about you, but the first time I saw the way that this works I thought it was a little like cheating… the instruction (the bits S1S0) didn’t really tell the computer what to compute. Rather, the computer just went ahead and computed everything (NOT A, A OR B, A AND B and A PLUS B) and then the instruction bits were just used to pick the desired function as the output.

While that seems wasteful, it turns out that is in fact the way the CPU in your current machine operates as well. The reason being is that it is less complex to build it that way than to build it so that it only does the desired operation. This is actually the case for lots of things in computing (and we’ll see more later). Further, not only is it less complex, but it is also faster.

A good analogy to see how it is faster right now is to think about the way that COVID vaccines were created. Many of the possible vaccines were mass produced before we even knew which, if any, of them would work. The idea being that if one or more did work, then since we produced them ahead of time, we only needed to pick the one(s) that did. That way, the vaccines that worked were ready to be distributed much more quickly (like the chosen output), and those that didn’t work were just ignored (like the other outputs).

Optional: To help me improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this Lab (include the 2-hour lab period in your total)?

b. Please comment on any particular challenges you faced in completing this Lab.

**Chip Pin-Out Reference:**

A picture containing clock

Description automatically generated