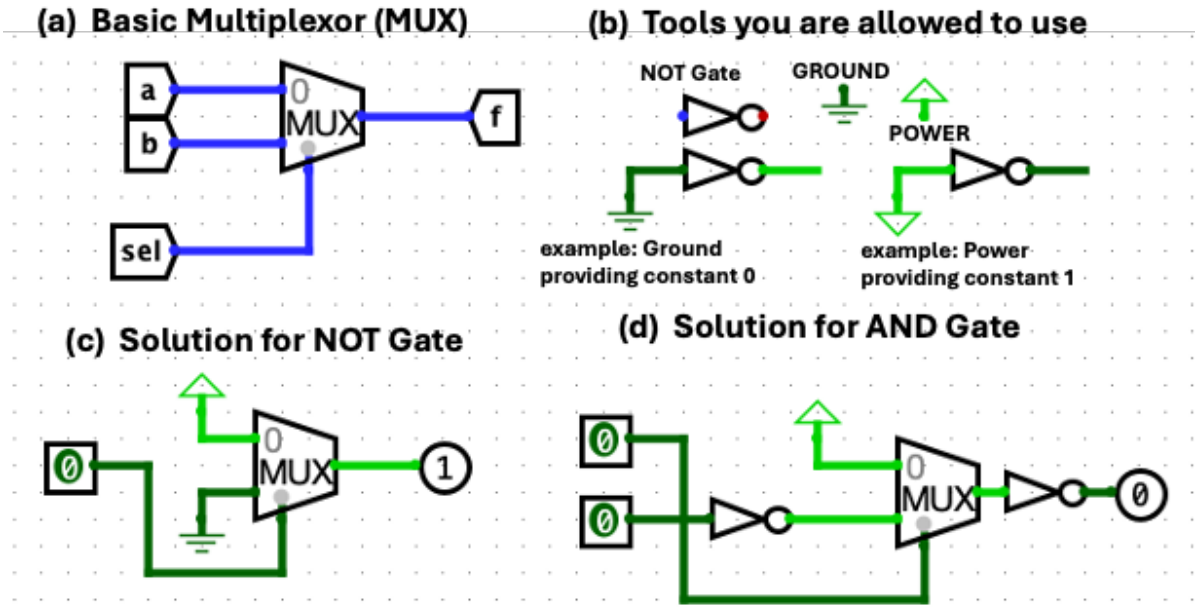
CS281 – Homework #4

1. We will be using multiplexors (mux) for a variety of purposes throughout the remainder of this term. For this problem we will have some "fun" constructing gates out of a multiplexor. Below (a) you see the symbol for the mux which consists of two inputs **a** and **b** and a selector **`sel`**, based on the value of **`sel`**, the mux produces an output **f** based on the inputs **a** and **b**.



Shown in the figure under (b) are the tools that you are allowed to use for this question. You may use wires, not gates, power and ground controls. Note that the power control is used to produce a constant value of 1 and the ground control is used to produce a constant value of 0. You may only use wires, not gates, power and ground to answer this question.

In class we examined the truth table for the **NOT**, **AND**, **OR**, **XOR**, **NAND**, **NOR**, and **XNOR** logic gates (see the slide). For this question, please design circuits to reproduce the correct logic behavior of the **OR**, **XOR**, **NAND**, **NOR**, and **XNOR** logic gates using only the tools allowed (b) and a single mux. To help you get started I have provided solutions for the **NOT** (c) and **AND** (d) gates in the figure above. You may hand draw your solutions (be neat), or use Logisim and even validate the correct operation (better).

Please only hand in an image of your design in either case (not a logisim file) – screen prints are fine to be pasted into your submission.

2. Assume that X consists of 3 bits, x_2, x_1, x_0. Write four logic functions that are true if and only if
   - $f_1(x)$: x contains only one 0
   - $f_2(x)$: x contains an even number of 0s (for this consider 0 an even number)
   - $f_3(x)$: x when interpreted as an unsigned binary number is less than 4
   - $f_4(x)$: x when interpreted as a signed (two's complement) number is negative

   First start by writing out the truth table for this question. The columns in the truth table should be x_2, x_1, x_0, f_1, f_2, f_3, and f_4. Use disjunctive normal form (sum of products) notation to describe the logic for each function, for example f_1= ??????, f_2= ??????, f_3= ??????, f_4= ??????