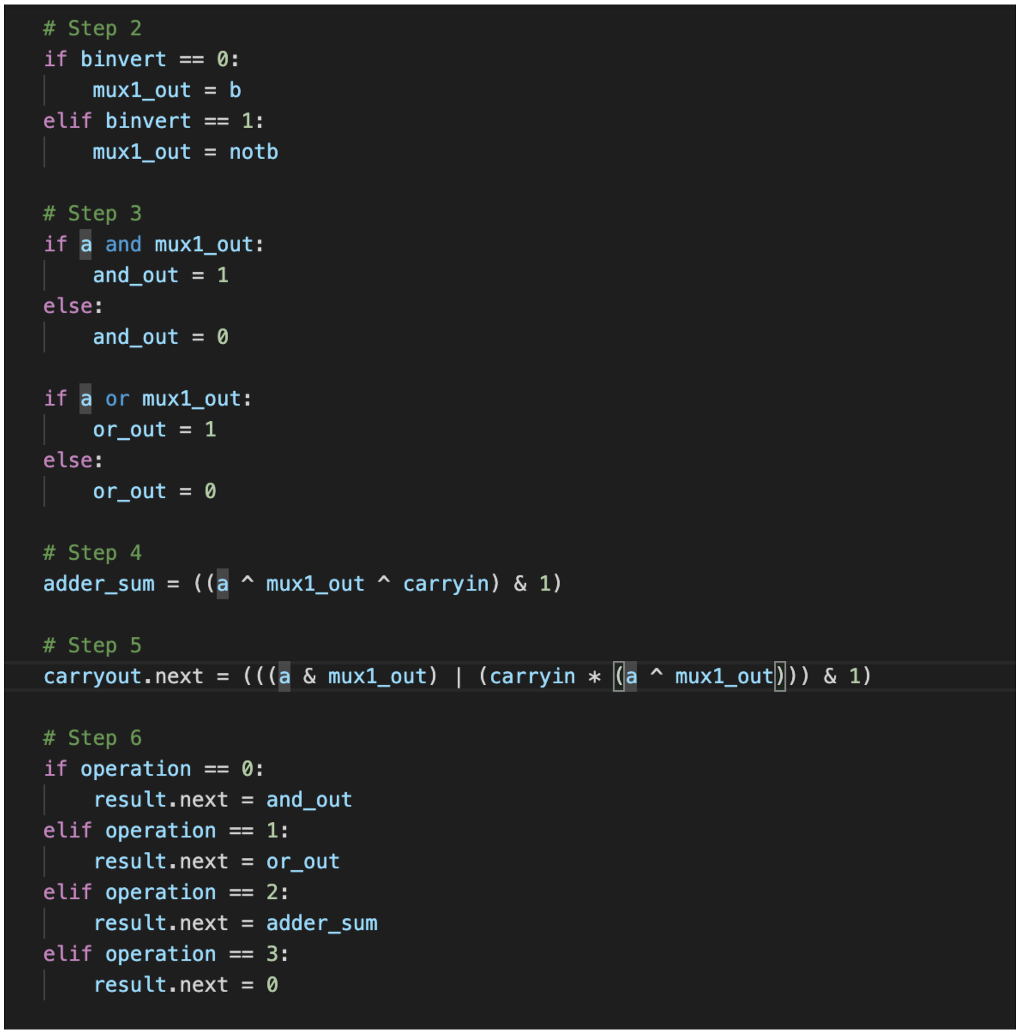
CSE 3666 - Lab 5
Jonathan Ameri

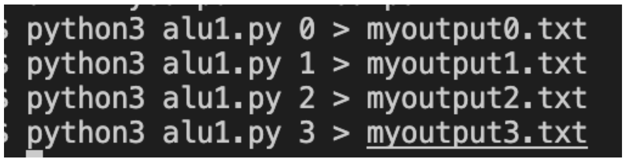
Submit a report in PDF format. The report should have the following.
Your work for each Step.



```
# Step 2
if binvert == 0:
    mux1_out = b
elif binvert == 1:
    mux1_out = notb

# Step 3
if a and mux1_out:
    and_out = 1
else:
    and_out = 0

if a or mux1_out:
    or_out = 1
else:
    or_out = 0

# Step 4
adder_sum = ((a ^ mux1_out ^ carryin) & 1)

# Step 5
carryout.next = (((a & mux1_out) | (carryin * (a ^ mux1_out))) & 1)

# Step 6
if operation == 0:
    result.next = and_out
elif operation == 1:
    result.next = or_out
elif operation == 2:
    result.next = adder_sum
elif operation == 3:
    result.next = 0
```

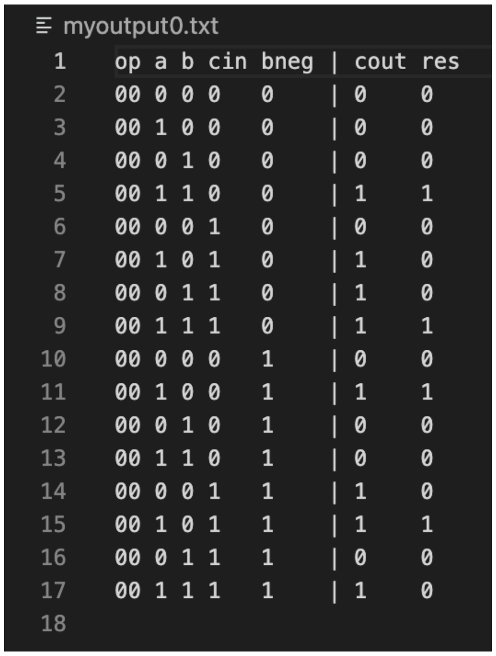
Screen shots of output of the program when operation is set to each of 0, 1, 2, and 3. Explain why you think the output is correct. Do NOT simply say it is the same as the provided output.

The following commands were run:



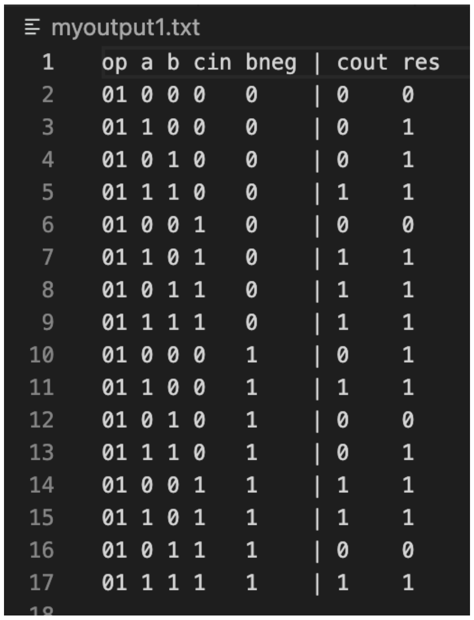
```
$ python3 alu1.py 0 > myoutput0.txt
$ python3 alu1.py 1 > myoutput1.txt
$ python3 alu1.py 2 > myoutput2.txt
$ python3 alu1.py 3 > myoutput3.txt
```

The outputs are as follows:



```
myoutput0.txt
op a b cin bneg | cout res
00 0 0 0   0    | 0    0
00 1 0 0   0    | 0    0
00 0 1 0   0    | 0    0
00 1 1 0   0    | 1    1
00 0 0 1   0    | 0    0
00 1 0 1   0    | 1    0
00 0 1 1   0    | 1    0
00 1 1 1   0    | 1    1
00 0 0 0   1    | 0    0
00 1 0 0   1    | 1    1
00 0 1 0   1    | 0    0
00 1 1 0   1    | 0    0
00 0 0 1   1    | 1    0
00 1 0 1   1    | 1    1
00 0 1 1   1    | 0    0
00 1 1 1   1    | 1    0
```

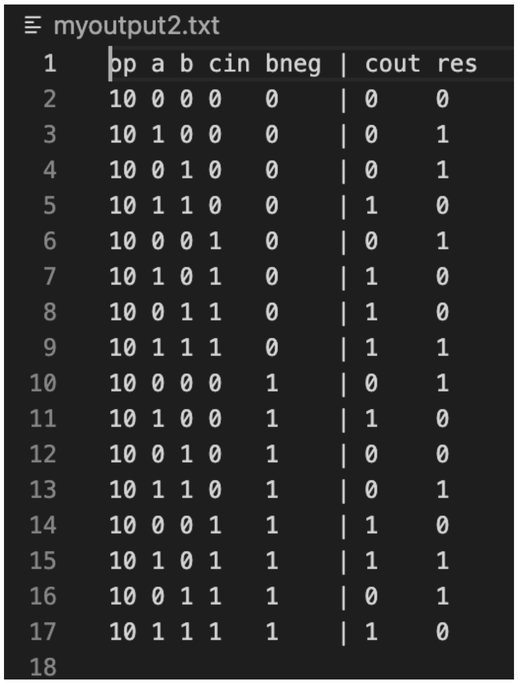
The output is correct. Since the operation bit is 0, we should have an output that correlates to the and function inside the ALU. If we look at the ‘res’ column, the result is only 1 when both a and b are 1 or when a is 1, b is 0, and bneg is 1. If we look at the ‘cout’ column, the output is still correct even though we are not using the ‘addition’ function. We have a 1 in cout when at least two of the three inputs a b and c are 1 or when bneg is 1, if at least two of the following are 1: a, notb, or cin.



```
myoutput1.txt
op a b cin bneg | cout res
01 0 0 0   0    | 0    0
01 1 0 0   0    | 0    1
01 0 1 0   0    | 0    1
01 1 1 0   0    | 1    1
01 0 0 1   0    | 0    0
01 1 0 1   0    | 1    1
01 0 1 1   0    | 1    1
01 1 1 1   0    | 1    1
01 0 0 0   1    | 0    1
01 1 0 0   1    | 1    1
01 0 1 0   1    | 0    0
01 1 1 0   1    | 0    1
01 0 0 1   1    | 1    1
01 1 0 1   1    | 1    1
01 0 1 1   1    | 0    0
01 1 1 1   1    | 1    1
```

This output is correct because the operation is 01 and the outputs of res follow the ‘or’ function of our ALU. When either a or b is 1, res is 1. Or when bneg is 1 and either a or notb is 1, res is 1.

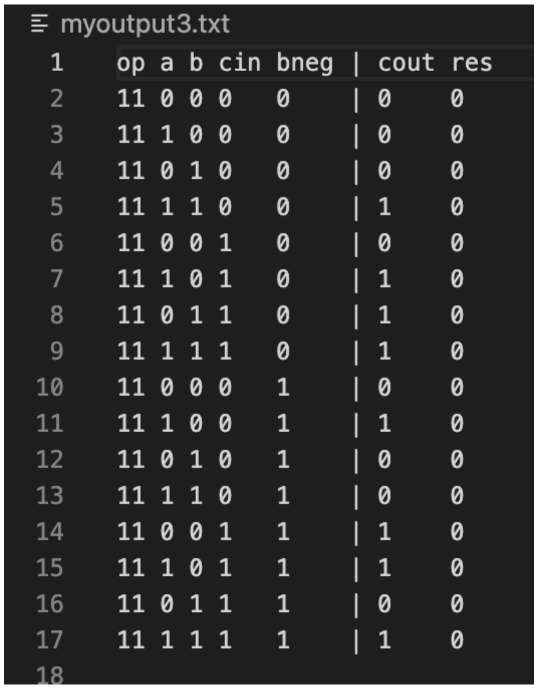
The ‘cout’ column is also correct even though we are not using the addition function. The reasoning behind this is explained in the previous answer.



myoutput2.txt

```
op a b cin bneg | cout res
10 0 0 0   0    | 0    0
10 1 0 0   0    | 0    1
10 0 1 0   0    | 0    1
10 1 1 0   0    | 1    0
10 0 0 1   0    | 0    1
10 1 0 1   0    | 1    0
10 0 1 1   0    | 1    0
10 1 1 1   0    | 1    1
10 0 0 0   1    | 0    1
10 1 0 0   1    | 1    0
10 0 1 0   1    | 0    0
10 1 1 0   1    | 0    1
10 0 0 1   1    | 1    0
10 1 0 1   1    | 1    1
10 0 1 1   1    | 0    1
10 1 1 1   1    | 1    0
```

The output here is correct. The operation is set to 10, which corresponds to the addition function. The result is 1 and cout is 0 when 1 of the following inputs are 1: a, b, and cin. When 2 of the inputs are 1, there is an overflow, so cout is 1 and res is 0. When all three inputs are 1, cout is 1 and res is 1, indicating that there is a double overflow. When bneg is 1, we apply all the same logic except using the inverse of b.



myoutput3.txt

```
op a b cin bneg | cout res
11 0 0 0   0    | 0    0
11 1 0 0   0    | 0    0
11 0 1 0   0    | 0    0
11 1 1 0   0    | 1    0
11 0 0 1   0    | 0    0
11 1 0 1   0    | 1    0
11 0 1 1   0    | 1    0
11 1 1 1   0    | 1    0
11 0 0 0   1    | 0    0
11 1 0 0   1    | 1    0
11 0 1 0   1    | 0    0
11 1 1 0   1    | 0    0
11 0 0 1   1    | 1    0
11 1 0 1   1    | 1    0
11 0 1 1   1    | 0    0
11 1 1 1   1    | 1    0
```

The output is correct because according to the ALU diagram, when the operation is set to 11, res should be 0 regardless of the other inputs, and the cout column should follow the same output as all the other functions because it is calculated regardless of what function is used.