CSE 232 SPRING 2020

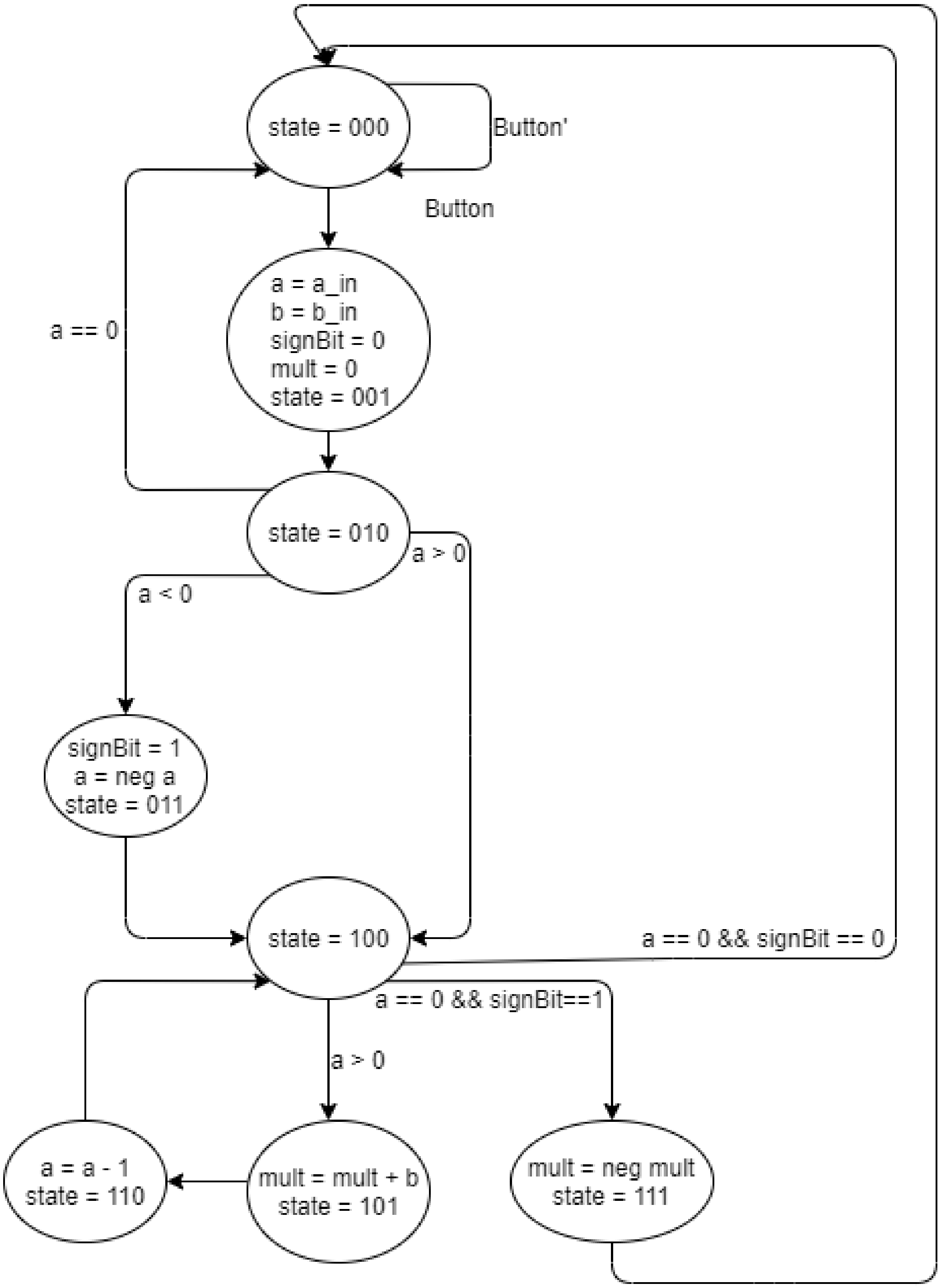
PROJECT 2

Ali Bahar-171044066

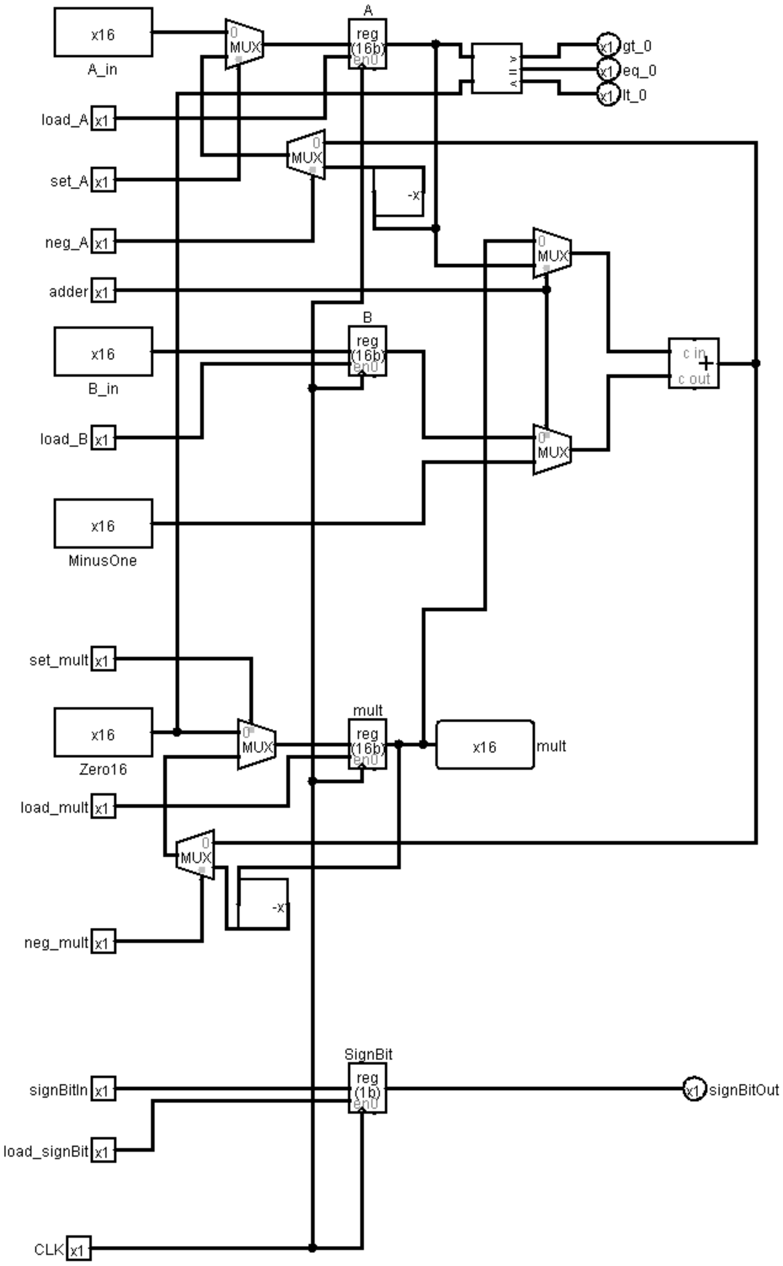
Problem solution with C

```c
signBit = 0;
a = a_in ;
b = b_in;
mult = 0;

if(a < 0){
    a = ~a;
    ++a;
    signBit = 1;
}

while(a > 0){
    mult = mult + b;
    a = a-1;
}

if(signBit){
    mult = ~mult;
    ++mult;
}

return mult;
```

State Diagram

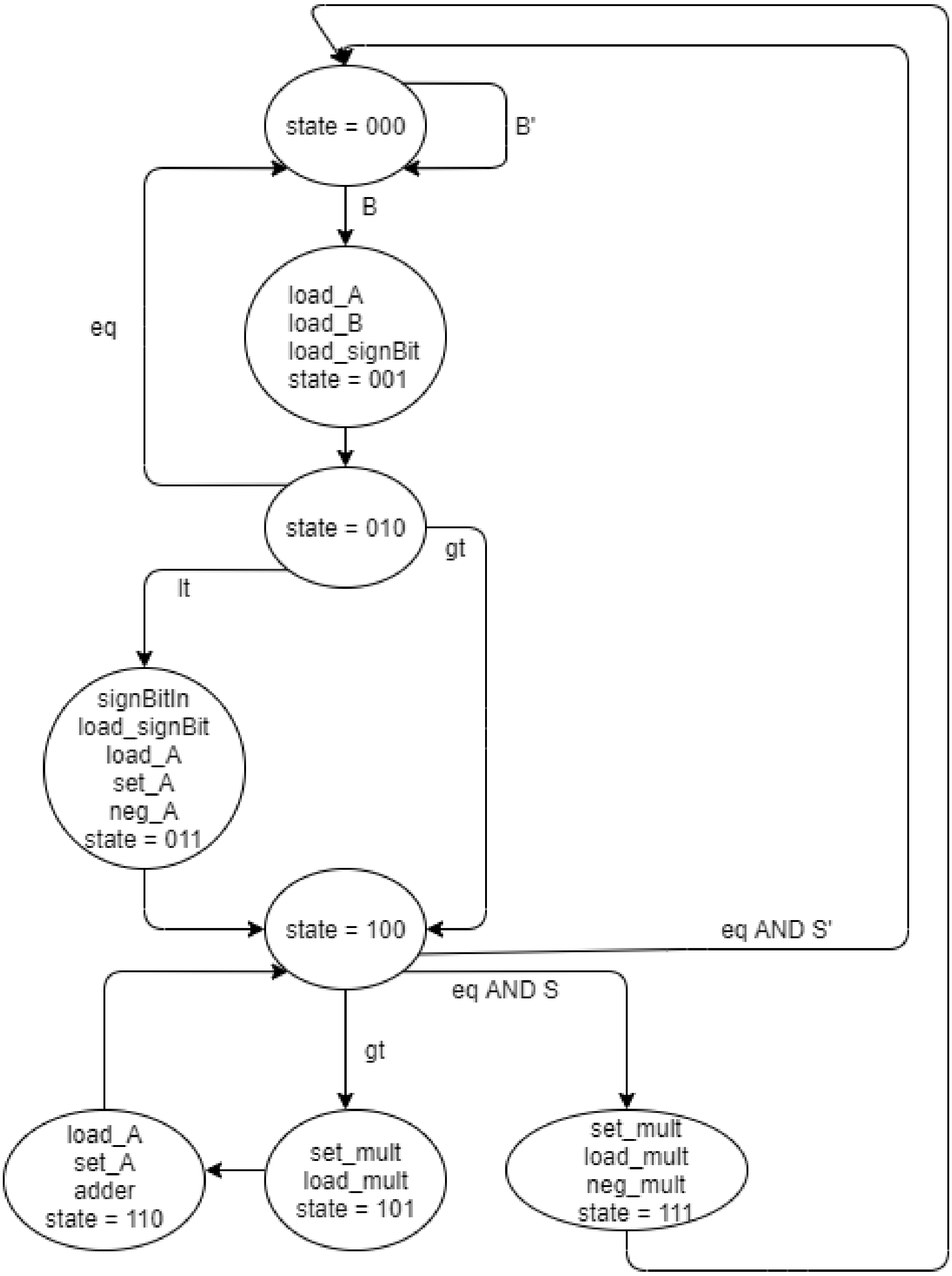


Data Path



x16
A_in
MUX
0
A
reg
(16b)
en0
>
=
<
x1
gt_0
eq_0
lt_0
load_A
set_A
neg_A
adder
-x
B
x16
B_in
load_B
c in
c out
+
x16
MinusOne
set_mult
x16
Zero16
mult
x16
mult
load_mult
neg_mult
SignBit
reg
(1b)
signBitIn
signBitOut
load_signBit
CLK

New State Diagram



Truth Table and Boolean Expressions

s2	s1	s0	gt	eq	ls	S	B	n2	n1	n0
0	0	0	x	x	x	x	0	0	0	0
0	0	0	x	x	x	x	1	0	0	1
0	0	1	x	x	x	x	x	0	1	0
0	1	0	1	0	0	x	x	1	0	0
0	1	0	0	1	0	x	x	0	0	0
0	1	0	0	0	1	x	x	0	1	1
0	1	1	x	x	x	x	x	1	0	0
1	0	0	1	0	0	x	x	1	0	1
1	0	0	0	1	0	1	x	1	1	1
1	0	0	0	1	0	0	x	0	0	0
1	0	1	x	x	x	x	x	1	1	0
1	1	0	x	x	x	x	x	1	0	0
1	1	1	x	x	x	x	x	0	0	0

s2=current state bit, s1=current state bit, s0=current state bit, gt=greater than, eq=equal, S=sign bit, n2=next state biy, n1=next state bit, n0=next state bit.

n2 = s2’.s1.s0’.gt + s2’.s1.s0 + s2.s1’.s0’.gt + s2.s1’.s0’ .eq.S + s2.s1’.s0 + s2.s1.s0’

n2 = s2’.s1.(gt + s0) + s2.s1’.s0’.(gt + eq.S) + s2(s1 XOR s0)

n1 = s2’.s1’.s0 + s2’s1.s0’.lt + s2.s1’.s0’.eq.S + s2.s1’.s0

n1 = s1’.(s0 + (s2.eq.S)) + s2’.s1.s0’.lt

no = s2’.s1’.s0’.B + s2’.s1.s0’.lt + s2.s1’.s0’.gt + s2.s1’.s0’.eq.S

n0 = s2’.s0’.(s1’.B + s1.lt) + s2.s1’.s0’.(gt + eq.S)

States	Load_A	Set_A	Neg_A	Load_B	Set_mult	Load_mult	Neg_mult	SignBitIn	Load_signBit	adder
S0 (000)	0	0	0	0	0	0	0	0	0	0
S1 (001)	1	0	0	1	0	1	0	0	1	0
S2 (010)	0	0	0	0	0	0	0	0	0	0
S3 (011)	1	1	1	0	0	0	0	1	1	0
S4 (100)	0	0	0	0	0	0	0	0	0	0

S5 (101)	0	0	0	0	1	1	0	0	0	0
S6 (110)	1	1	0	0	0	0	0	0	0	1
S7 (111)	0	0	0	0	1	1	1	0	0	0

Load_A = S1 + S3 + S6

Set_A = S3 + S6

Neg_A = S3

Load_B = S1

Set_mult = S5 + S7

Load_mult = S1 + S5 + S7

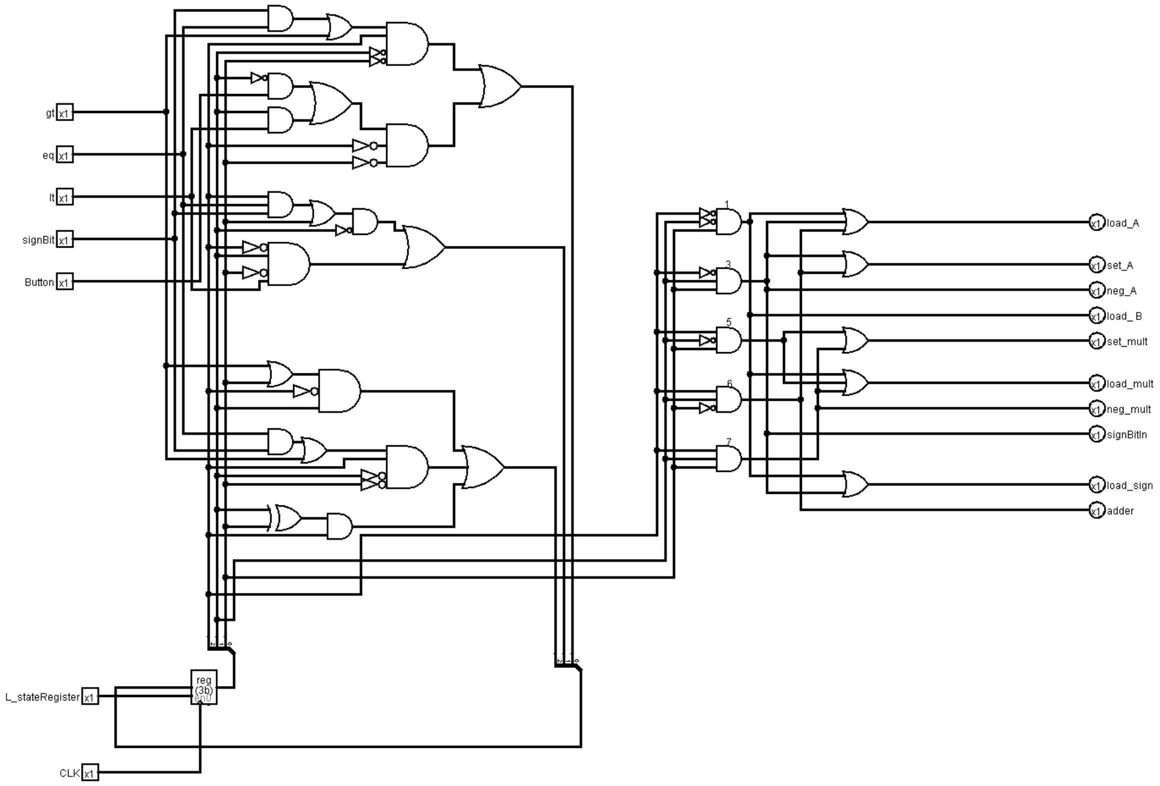
Neg_mult = S7

SignBitIn = S3

Load_signMit = S1 + S3

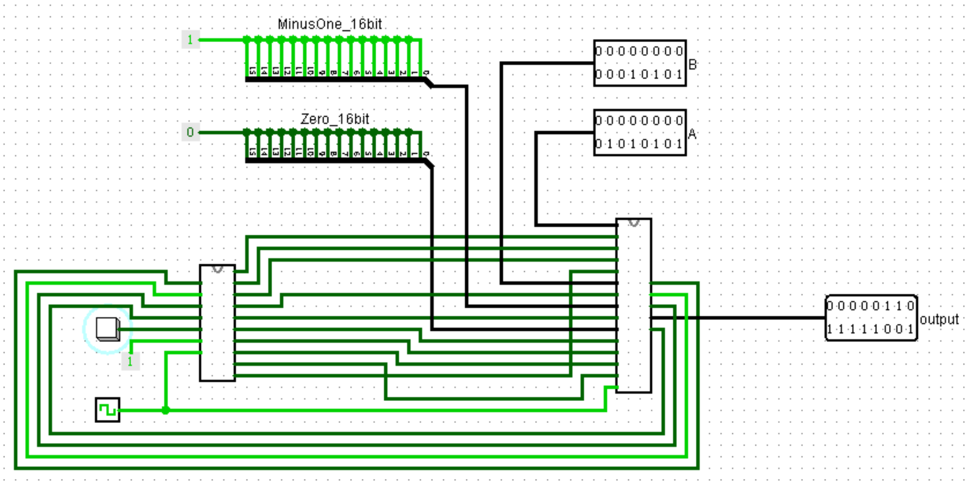
Adder = S6

Control Unit



Test

- **Case 1 (positive - positive)**

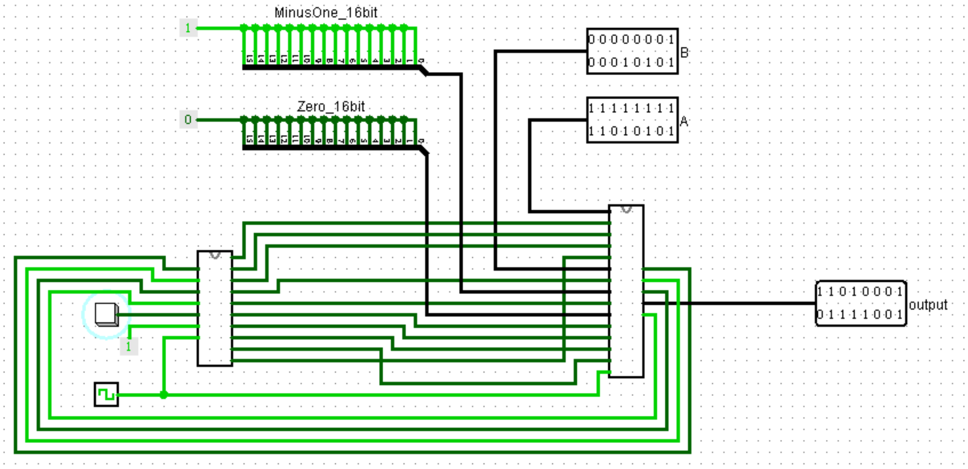


A = 00000000 00010101 = 21

B = 00000000 01010101 = 85

Output = 00000110 11111001 = 1785

- **Case 2 (negative - positive)**

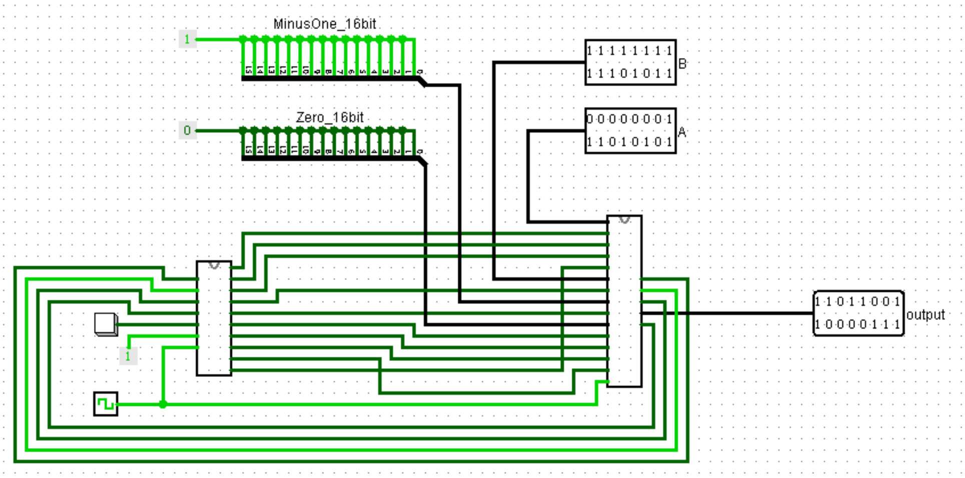


A = 11111111 11010101 = -43

B = 00000001 00010101 = 277

Output = 11010001 01111001 = -11911

- **Case 3 (positive- negative)**

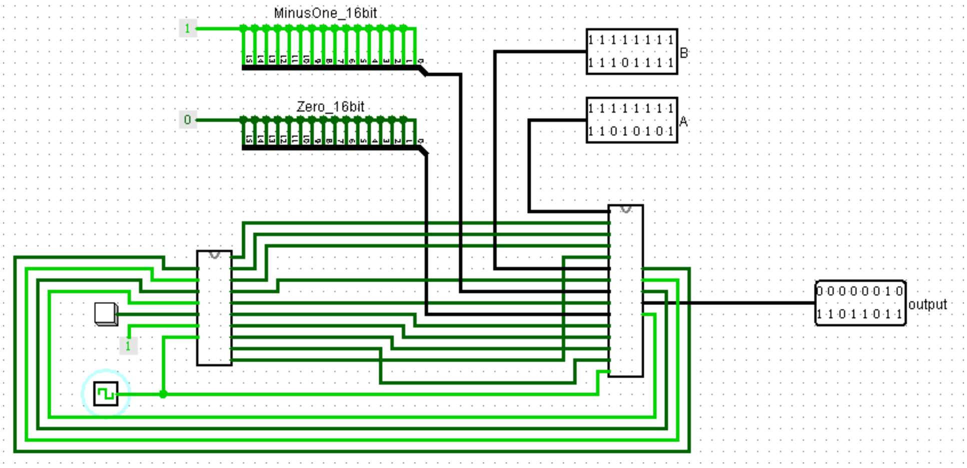


A = 00000001 11010101 = 469

B = 1111111 11101011 = -21

Output = 11011001 10000111 = -9849

- **Case 4(negative - negative)**



A = 11111111 11010101 = -43

B = 11111111 11101111 = -17

Output = 00000010 11011011 = 731