GIT Department of Computer Engineering
CSE 232 - Spring 2020
Project 2

BARAN HASAN BOZDUMAN
171044036

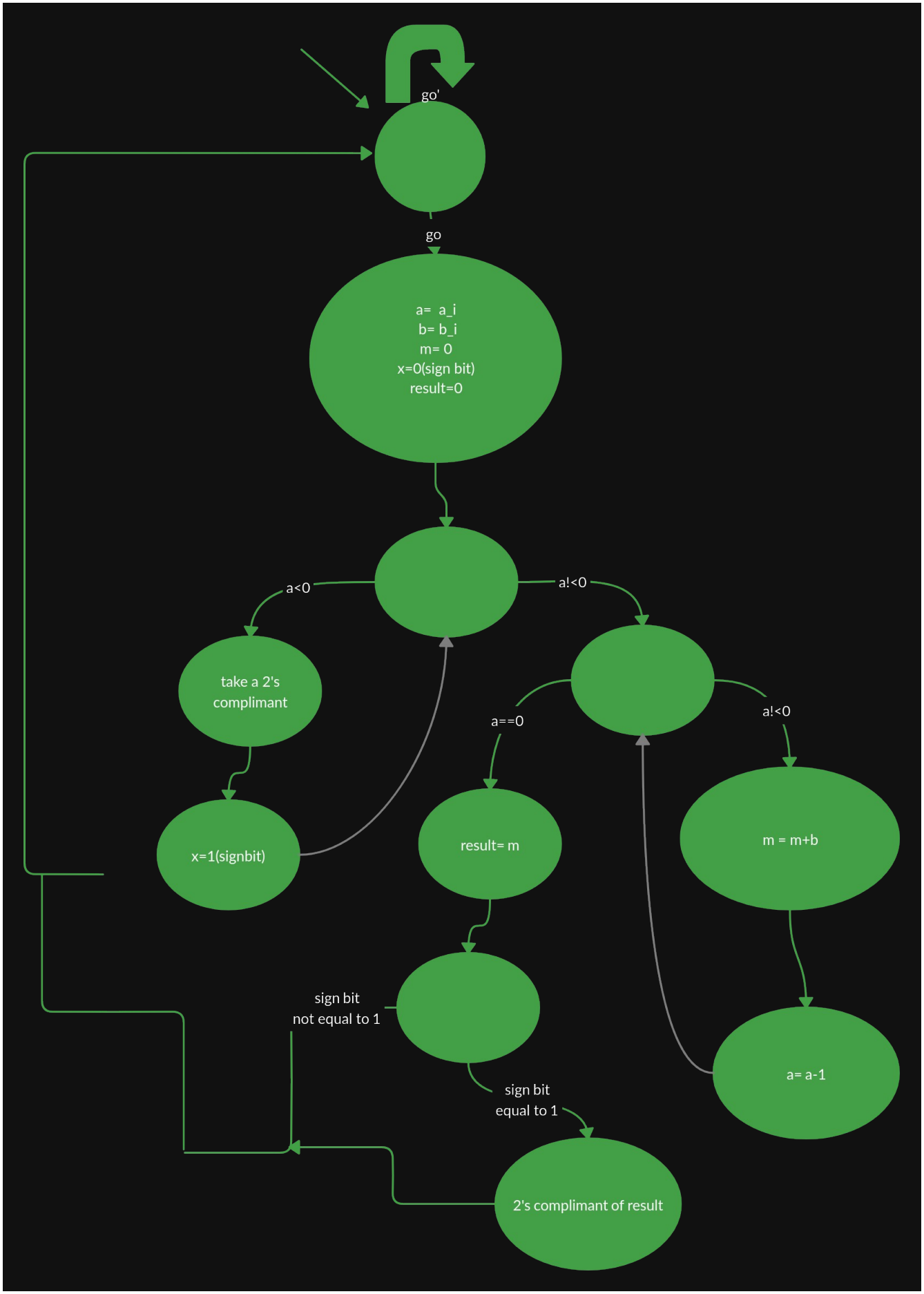
CODE OF MULTIPLICATION

```
mult = 0;

while( a > 0 ){

        mult = mult + b;

        a = a - 1;
}
```

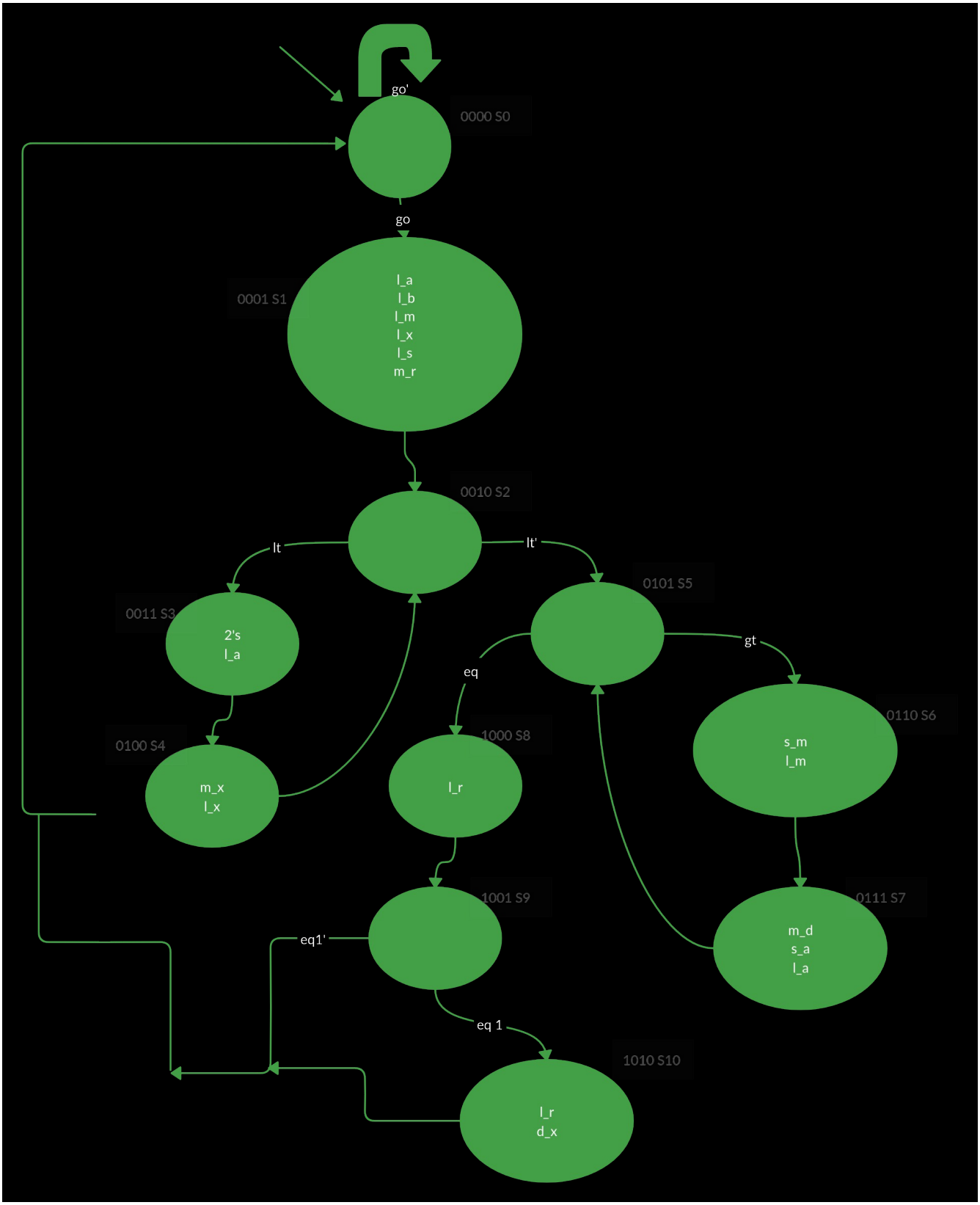
Its obvious the pogrem does multiplication by using adding operations it makes a timees add b to result.

DIAGRAM FOR DATAPATH

Firstly before we start to create circuit we need to create a data path and we does to cahangeable operations we connect them to mux and (for appropriate points for instance when you connect the wire which is you use often so you dont need to change it to one for each time)and if you assign some thing into a register after that state you compare it with something or change it to another value it behaves for old one so you need to put an empty statement beetween of them and it provides one more clock time.


go'
go
a= a_i
b= b_i
m= 0
x=0(sign bit)
result=0
a<0
a!<0
take a 2's complimant
x=1(signbit)
a==0
a!<0
result= m
m = m+b
sign bit not equal to 1
sign bit equal to 1
a= a-1
2's complimant of result

FINITE STATE MACHINE



I convert our operationns into proper boolean values and we give name to our states I use l_ for the load memory and m_ is for the cahnge mux es lt (lessthan)gt(greater then) etc.

TRUTH TABLE
(for state changes)

INPUTS									**OUTPUTS**			
P3	P2	P1	P0	lt	eq	eq1	go	gr	N3	N2	N1	N0
0	0	0	0	-	-	-	0	-	0	0	0	0
0	0	0	0	-	-	-	1	-	0	0	0	1
0	0	0	1	-	-	-	-	-	0	0	1	0
0	0	1	0	1	-	-	-	-	0	0	1	1
0	0	1	0	0	-	-	-	-	0	1	0	1
0	0	1	1	-	-	-	-	-	0	1	0	0
0	1	0	0	-	-	-	-		0	0	1	0
0	1	0	1	-	-	-	-	1	0	1	1	0
0	1	0	1	-	1	-	-	-	1	0	0	0
0	1	1	0	-	-	-	-	-	0	1	1	1
0	1	1	1	-	-	-	-	-	0	1	0	1
1	0	0	0	-	-	-	-	-	1	0	0	1
1	0	0	1	-	-	1	-	-	1	0	1	0
1	0	0	1	-	-	0	-	-	0	0	0	0
1	0	1	0	-	-	-	-	-	0	0	0	0

N3=P3’P2P1’P0eq +P3P2’P1’P0’+ P3P2’P1’P0eq1
simplified = P3P2’P1’(P0 +P0eq1) + P3’P2P1’P0eq
(P0’ + eq)

N2= P3’P2’P1P0’lt’ + P3’P2’P1P0 + P3’P2P1’P0gr + P3’P2P1P0’ + P3’P2 P1P0
simplified= P3’P1P0 + P3’P1lt’ + P3’P1P2 + P3’P2P0gr

N1=P3’P2’P1’P0+ P3’P2’P1P0’lt + P3’P2P1’P0’ + P3’P2P1’P0gr + P3’P2P1P0’ + P3P2’P1’P0eq1
simplified = P3’P1P0’lt + P3’P2’P1’P0 + P3’P2P1’gr + P3’ P2P0’+ P0P1’P2’eq1

N0 = P3’P2’P1’P0’go + P3’P2’P1P0’lt + P3’P2’P1P0’LT’ + P3’P2P1P0’ + P3’P2P1P0 +P3P2’P1’P0’
simplified =P3P2’P1’P0’ + P3’P1P0’ +P3’P2P1 + P2’P1’P0’go

TRUTH TABLE
(for state operations)

INPUTS					**OUTOUTS**										
	P3	P2	P1	P0	la	lb	lm	lx	dx	2s	lr	mx	sm	md	sa
S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
S2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
S3	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
S4	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0
S5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
S6	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0
S7	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1
S8	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
S9	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
S10	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0

La =S1 +S3 +S7
lb =S1
lm = S1 +S6
lx =S1 +S4
dx =S10
2s =S3
lr = S8 +S10
mx = S4
sm = S6
md =S7
sa =S7
mr =S1

note: My logisim circuit is working for negative numbers two just may be it will be give an overflow error if the number bigger than 16 bit but as teacher indicated it does not matter..