Don Bosco Institute of Technology,Kurla

NAME:Hayden Cordeiro **SECOMPS** **ROLLNO:05**

EXPERIMENT NO:3

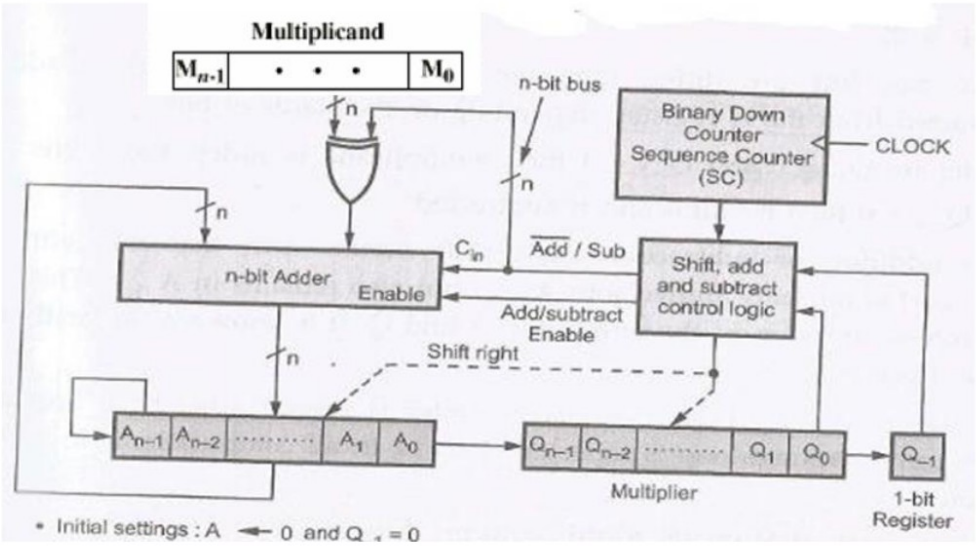
Title: A program to simulate Booth's multiplication

Class:S.E **Lecturer:Sejal.Ch**

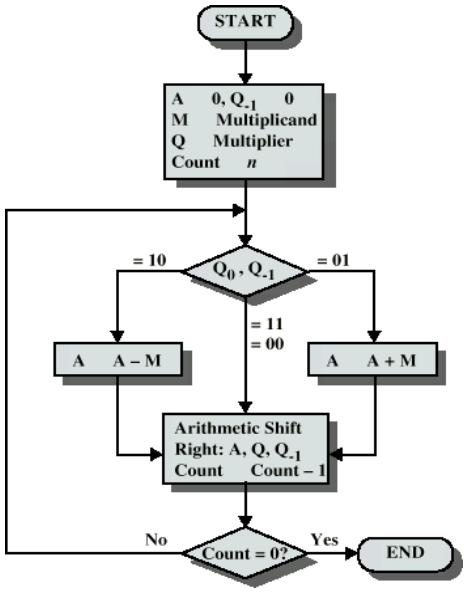
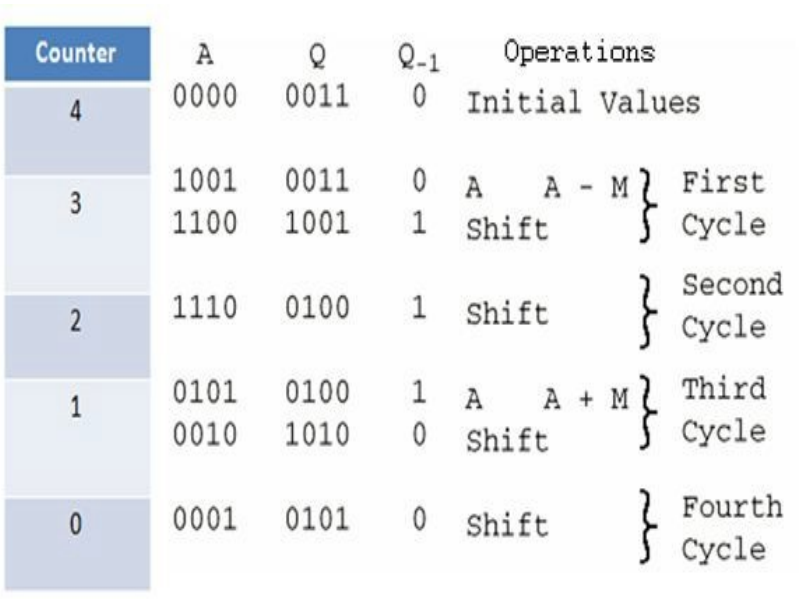
Subject: PA

EXPERIMENT NO: 3
Simulate Booth's Algorithm

AIM	Write a program to simulate Booth's multiplication
LEARNING OBJECTIVE	To implement the operation of the arithmetic unit including the implementation of fixedpoint multiplication for signed numbers.
LEARNING OUTCOME	Students will be able to write a higher level language code for simulating hardware operation for Booth's multiplication process.
LAB OUTCOME	CSL 403.1: Ability to compile a code for computer operations.
PROGRAM OUTCOME	PO11, PO52, PO83, PO93, PO122, PSO12
BLOOM'S TAXONOMY LEVEL	Remember, Understand
THEORY	Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m. As in all multiplication schemes, booth algorithm requires examination of the multiplier bits and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules: algorithm: 1.Take two decimal numbers from the user in the range of 0 to 15,both positive and negative(M=Multiplicand & Q=Multiplier)

	and convert it to 4- bit binary numbers. 2.Negative numbers are represented in two's complement form. 3.Intialise the counter with the count of number of bits . 4.Intialise the one bit register Q-1 as 0. 5.Intialise A to zero ,where A is accumulator which stores the MSB of the result . Multiplier Q stores the LSB of the result. 6.Check LSB bit of Q and Q-1 : a.If they are 11 or 00 arithmetic shift is done for A,Q and Q-1 and decrement the counter. b.If they are 01 add A to M and then arithmetic shift is done for A,Q and Q-1 and decrement the counter. c.If they are 10 sub M from A and then arithmetic shift is done for A,Q and Q-1 and decrement the counter. 7.Check the counter,if it is not 0,move to step 6,otherwise store the result in A and Q. 
SOFTWARE USED	C/C++/Java

STEPS TO EXECUTE THE PROGRAM	1. Take two decimal numbers from the user in the range of 0 to 15,both positive and negative(M=Multiplicand & Q=Multiplier) and convert it to 4 bit binary numbers. 2. Negative numbers are represented in two's complement form. 3.Intialise the counter with the count of number of bits . 4. Intialise the one bit register Q1 as 0. 5. Intialise A to zero ,where A is accumulator which stores the MSB of the result . Multiplier Q stores the LSB of the result. 6. Check LSB bit of Q and Q1 : a.If they are 11 or 00 arithmetic shift is done for A,Q and Q1 and

	decrement the counter. b. If they are 01 add A to M and then arithmetic shift is done for A,Q and Q1 and decrement the counter. c. If they are 10 sub M from A and then arithmetic shift is done for A,Q and Q1 and decrement the counter. 7.Check the counter,if it is not 0,move to step 6,otherwise store the result in A and Q.
FLOWCHART	
EXAMPLE	
CODE	3A:

Counter	A	Q	Q_{-1}	Operations	
4	0000	0011	0	Initial Values	
3	1001	0011	0	A A - M	First Cycle
	1100	1001	1	Shift	
2	1110	0100	1	Shift	Second Cycle
1	0101	0100	1	A A + M	Third Cycle
	0010	1010	0	Shift	
0	0001	0101	0	Shift	Fourth Cycle

```
m=int(input())
md=int(input())
a='00000'
qm1='0'
l=1
d=bin(int(l))[2:].zfill(5)
counter=4
if(m<0):
        print("m is lesser theab 0")
        m*=-1
        c=bin(m)[2:].zfill(5)
        c = c.replace('0', '.')
        c = c.replace('1', '0')
        c = c.replace('.', '1')
        m=str(bin(int(c,2) + int(d,2))[2:]).zfill(5)

else:
        m=bin(m)[2:].zfill(5)


if(md<0):
        l=1
        md*=-1
        c=bin(md)[2:].zfill(5)
        c = c.replace('0', '.')
        c = c.replace('1', '0')
        c = c.replace('.', '1')

        md=str(bin(int(c,2) + int(d,2))[2:]).zfill(5)


else:
        md=bin(md)[2:].zfill(5)


c=m
c = c.replace('0', '.')
c = c.replace('1', '0')
c = c.replace('.', '1')
print("c {}".format(c))
twos=str(bin(int(c,2) + int(d,2))[2:]).zfill(5)

counter=len(md)-1
print("a\t md \t q-1 \t counter\toperartion")
print("{}  {}     {}           {}
initial".format(a,md,qm1,counter+1))
op1=md[-1]
```

```
op2=qm1
while(counter!=-1):
	if(  op1=='0' and op2=='0'):

		qm1=md[-1]
		temp=md;
		md=a[-1]+temp
		md=md[0:len(md)-1]
		a=a[0]+a
		a=a[0:len(a)-1]

		print("{}  {}    {}          {}     ARITHMETIC
RIGHT SHIFT".format(a,md,qm1,counter))


	if(  op1=='1' and op2=='1'):

		qm1=md[-1]
		temp=md;
		md=a[-1]+temp
		md=md[0:len(md)-1]
		a=a[0]+a
		a=a[0:len(a)-1]
		print("{}  {}    {}          {}     ARITHMETIC
RIGHT SHIFT".format(a,md,qm1,counter))


	if(  op1=='1' and op2=='0' ):

		a=str(bin(int(a,2) + int(twos,2))[2:])
		a=a.zfill(5)
		a=a[-5:]
		print("{}  {}    {}          {}     A-
B".format(a,md,qm1,counter))
		qm1=md[-1]
		temp=md;
		md=a[-1]+temp
		md=md[0:len(md)-1]
		a=a[0]+a
		a=a[0:len(a)-1]
		print("{}  {}    {}          {}     ARITHMETIC
RIGHT SHIFT".format(a,md,qm1,counter))



	if(op1=='0' and op2=='1'):

		a=str(bin(int(a,2) + int(m,2))[2:])
```

```
            a=a.zfill(5)

      #     a=a[1:6]
            a=a[-5:]
            print("{}  {}    {}        {}
A+B".format(a,md,qm1,counter))
            qm1=md[-1]
            temp=md;
            md=a[-1]+temp
            md=md[0:len(md)-1]
            a=a[0]+a
            a=a[0:len(a)-1]
            print("{}  {}    {}        {}     ARITHMETIC
RIGHT SHIFT".format(a,md,qm1,counter))


      counter-=1
      op1=md[4]
      op2=qm1
```

cases:

```
-7
-2
c 00110
a        md      q-1     counter        operartion
00000  11110     0            5       initial
00000  01111     0            4       ARITHMETIC RIGHT SHIFT
00111  01111     0            3       A-B
00011  10111     1            3       ARITHMETIC RIGHT SHIFT
00001  11011     1            2       ARITHMETIC RIGHT SHIFT
00000  11101     1            1       ARITHMETIC RIGHT SHIFT
00000  01110     1            0       ARITHMETIC RIGHT SHIFT
```

```
7
2
c 11000
a        md      q-1     counter        operartion
00000  00010     0            5       initial
00000  00001     0            4       ARITHMETIC RIGHT SHIFT
11001  00001     0            3       A-B
11100  10000     1            3       ARITHMETIC RIGHT SHIFT
00011  10000     1            2       A+B
00001  11000     0            2       ARITHMETIC RIGHT SHIFT
00000  11100     0            1       ARITHMETIC RIGHT SHIFT
00000  01110     0            0       ARITHMETIC RIGHT SHIFT
```

```
-7
2
c 00110
a        md      q-1     counter        operartion
00000  00010     0             5      initial
00000  00001     0             4      ARITHMETIC RIGHT SHIFT
00111  00001     0             3      A-B
00011  10000     1             3      ARITHMETIC RIGHT SHIFT
11100  10000     1             2      A+B
11110  01000     0             2      ARITHMETIC RIGHT SHIFT
11111  00100     0             1      ARITHMETIC RIGHT SHIFT
11111  10010     0             0      ARITHMETIC RIGHT SHIFT
```

```
2
-7
c 11101
a        md      q-1     counter        operartion
00000  11001     0             5      initial
11110  11001     0             4      A-B
11111  01100     1             4      ARITHMETIC RIGHT SHIFT
00001  01100     1             3      A+B
00000  10110     0             3      ARITHMETIC RIGHT SHIFT
00000  01011     0             2      ARITHMETIC RIGHT SHIFT
11110  01011     0             1      A-B
11111  00101     1             1      ARITHMETIC RIGHT SHIFT
11111  10010     1             0      ARITHMETIC RIGHT SHIFT
```

3B:

```
a=int(input())
lis=[]
if(a<0):
        a*=-1
        a=bin(a)[2:]
        a="1"+a+"0"
        print(a)
else:
        a=bin(a)[2:]
        a="0"+a+"0"


def compare(a,b):
        if(a=='0' and b=='1'):
                return 1
        elif(a=='1' and b=='1'):
                return 0
        elif(a=='1' and b=='0'):
                return -1
```

```
for i in range(1,len(a)):
        lis.append(int(compare(a[i-1],a[i])))

print(*lis)

for i in range(0,len(a)-1,2):
        print(2*lis[i]+lis[i+1] ,end=" ")
```

```
-27
1110110
0 0 -1 1 0 -1
0 -1 -1 (base)
```

```
+27
1 0 -1 1 0 -1
2 -1 -1 (base)
```

CONCLUSION	We have successfully implemented booths algorithm and carried out signed multiplication . Also several test cases were taken and verified . Extra scope of bit pair re encoding was also successfully implemented
REFERENCES	1. William Stallings, “Computer Organization and Architecture: Designing for Performance”,Pearson Publication, 10 th Edition, 2013 2. B. Govindarajulu, “Computer Architecture and Organization: Design Principles and Applications”, Second Edition, McGrawHill (India)