TOE BAI10381
Mijmal Jajru
SCSE (AI and ML)

Booths Algorithm

Booths algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in efficient way, i.e, less number of addition Substraction required.

Booth algorithm includes shifting. ~~the~~ ~~rrrr~~ prior to the shifting, the multiplicand may be added to the partial product. Substract from the partial product, or left unchanged according to following Rules:

1) The multiplicand is subtracted from the partial product upon encountering the first least. Significant 1 in a string of 1's in the multiplier
2) The multiplicand is added to the partial product upon encountering the first 0 (provide that there was a previous) in a string of 0's in the multiplier
3) The partial product does not change when the

multiplier bit is identical to the previous multiplier bit.

Example ⟶

Q	Q_{n-1}		
1	0	⟶	AC = AC − M ⟶ ASR
0	1	⟶	AC = AC + M ⟶ ASR
0	0	⟶	ASR
1	1	⟶	ASR

Ex ⇒ Multiply 7 and 3 using Booth's algorithm.

⟶ Register → 4

M → $(7)_{10} = (0111)_2$

Q → $(3)_{10} = (0011)_2$

$(-m) \rightarrow (1001)_2$

AC = AC − M

AC = AC + (−m)

−m = 2's

0 1 1 1

(1's) 1 0 0 0

(1's + 1) 0 0 0 1

2's 1 0 0 1

① AC	Q	Q_{n-1}	operation
0 0 0 0	0 0 1 (1)	(0)	i) AC = AC − M
			0 0 0 0
			+ 1 0 0 1
			1 0 0 1
1 0 0 1	0 0 1 1	0	ii) A.S.R
1 1 0 0	1 0 0 (1)	(1)	
② 1 1 0 0	1 0 0 (1)	(1)	i) A.S.R
1 1 1 0	0 1 0 0	1	

④ 0 1 0 1 0 1 0 0 1

0 0 1 0 1 0 1 0 0

i) $Q_0 \cdot Q_{-1} = 01$

$Ac = Ac + M$

$$\begin{array}{r} 1110 \\ +\ 0111 \\ \hline 0101 \end{array}$$

ii) A·S·R.

⑤ 0 0 0 1 0 1 0 1 ~~0~~

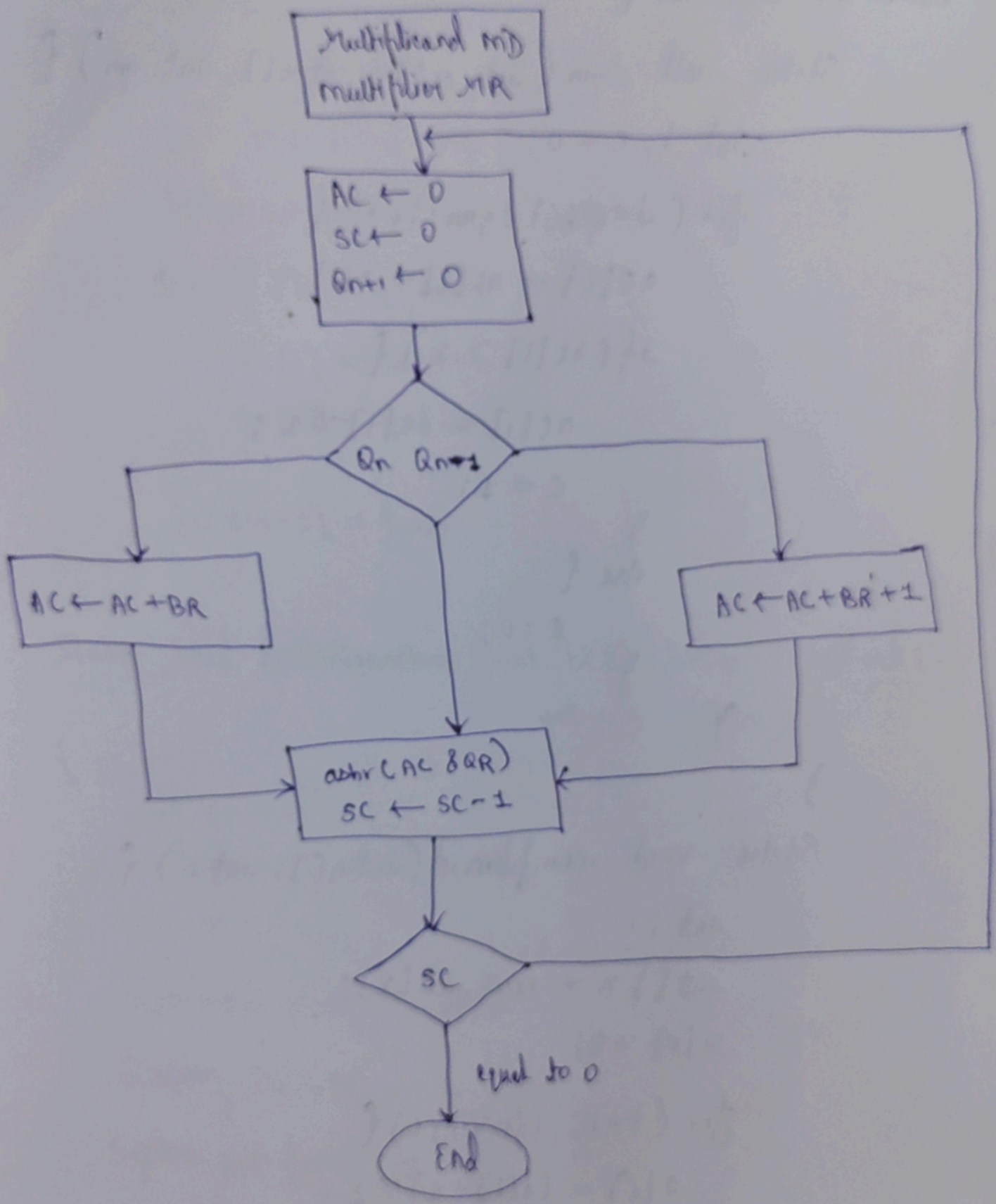
$(0\ 0\ 0\ 1\ 0\ 1\ 0\ 1)_2 = (21)_{10}$

New Examples → 5 X 4

The multiplicand is 5 ⇒ 0101

A	Q	Q_{n-1}	M	operation
0000	0100	0	0101	Initialize data
0000	0010	0	0101	shift
0000	0001	0	0101	shift
1011	0001	0	0101	A = A - M
1101	1000	1	0101	shift
0010	1000	1	0101	A = A + M
0001	0100	0	0101	shift

Booth Algorithms, flow chart →



Booth Code →

```
class Boothalgorithm {
    Static void Sum (int ac[], int x[], int qn) {
        int i, c=0;
        for (i=0; i<qn; i++) {
            ac[i] = ac[i] + x[i] + c;
            if(ac[i] > 1) {
                ac[i] = ac[i] % 2;
                c = 1;
            }
            else {
                c=0;
            }
        }
    }
    Static void complement (int a[], int n) {
        int i;
        int[] x = new int[8];
        x[0] = 1;
        for (i=0; i<n; i++) {
            a[i] = (a[i]+1) % 2;
        }
        Sum( a, x, n);
    }
```

```
Static void rightshift( int ac[], int qr[], int qn , int qrn)
{
    int temp , i ;
    temp = ac [0];
    qn = qr [0];
    System.out.print(" right shift");
    for (i=0; i<qrn-1 ; i++){
        ac[i] = ac[i+1];
        qr[i] = qr[i+1];
    }
    qr[qrn-1] = temp;
}
Static Void boothAlgorithm ( int br[], int qr[], int mt[],
                             int qrn, int sc)
{
    int qn=0;
    int[] ac = new int [10];
    int temp=0;
    System.out.print (" initial ");
    display (ac, qr, qrn);
    System.out.print( sc );
    while (sc!=0) {
        System.out.print( qr[0] + "\t" + qn);
        if ((qn + qr[0]) == 1)
```

```
{
  if (temp == 0)
  {
    add (ac, mt, qrn);
    System.out.print("A = A - Br");
    for (int i = qrn - 1; i >= 0; i--)
    {
      System.out.print(ac[i]);
    }
    temp = 1;
  }
  else if (temp == 1)
  {
    add (ac, br, qrn);
    System.out.print (A = A + BR)
    for (int i = qrn - 1; i >= 0; i--)
    {
      system.out.print (ac[i]);
    }
    temp = 0;
  }
  System.out.print();
  rightshift (ac, qr, qn, qrn);
}
else if (qn - qr[0] == 0)
{ rightshift (ac, qr, qn, qrn);
}
```

```
        System.out.print("\t");
        sc--;
        System.out.print("t"+sc+"\n");
    }
}
static void reverse(int a[]){
    int i, k, n=a.length;
    int t;
    for(i=0; i<n/2; i++)
    {
        t=a[i];
        a[i]=a[n-i-1];
        a[n-i-1]=t;
    }
}
public static void main(String[] args)
{
    int[] mt = new int[10];
    int sc;
    int brn; qrn;
    brn = 4;
    int br[] = {0,1,1,0};
    for(int i=brn-1; i>=0; i--)
    {   mt[i] = br[i];
    }
```

```
reverse(br);
Complement(mt, brn);
qrn = 4;
sc = qrn;
int qr[] = {1, 0, 1, 0};
reverse(qr);
boothAlgorithm(br, qr, mt, qrn, sc);
System.out.print("Result");
for(int i = qrn-1; i >= 0; i--){
    System.out.print(qr[i]);
}
}
}
```