BOOTH'S ALGORITHM

Contents

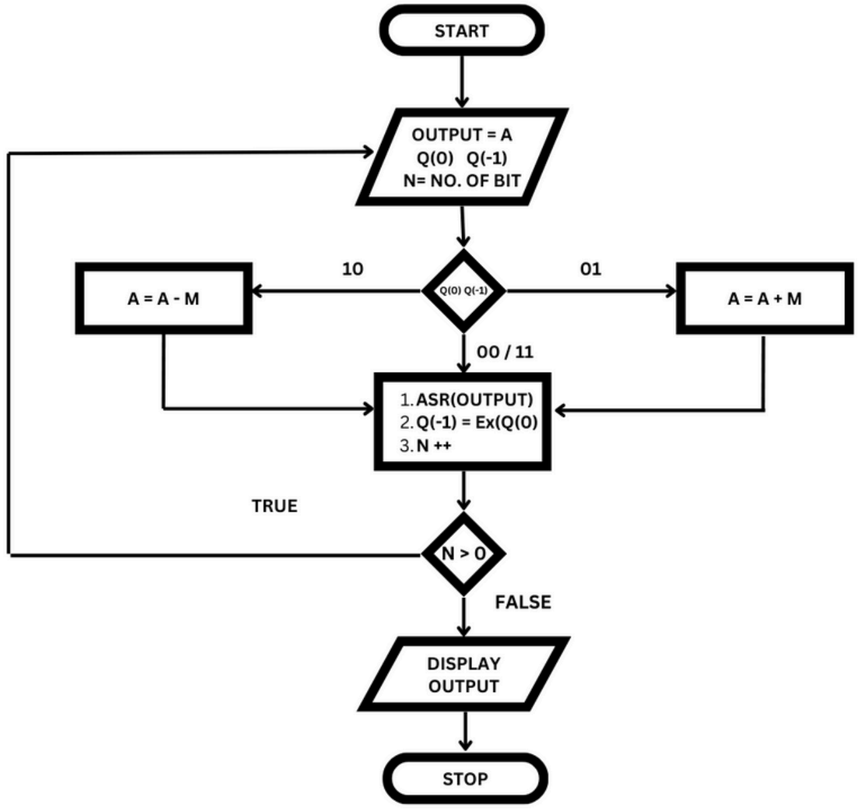
Introduction

In the world of computing, multiplication is crucial for performing complex calculations quickly, which is essential for everything from simple tasks like displaying graphics to more advanced applications like cryptography and simulations. To perform multiplication in utilizing minimal number of steps and less time complexity we use Booth’s multiplication algorithm. Booth's Algorithm, a method for binary multiplication, streamlines calculations by minimizing shifts and additions, particularly efficient for numbers with repetitive 1s or 0s.

Algorithm

STEP 1:Initialization
STEP 2:Iterate through Multiplier Bits
STEP 3:Shift and Count
STEP 4:Loop Termination

Flow chart


START
OUTPUT = A
Q(0) Q(-1)
N= NO. OF BIT
10
Q(0) Q(-1)
01
A = A - M
A = A + M
00 / 11
1. ASR(OUTPUT)
2. Q(-1) = Ex(Q(0)
3. N ++
TRUE
N > 0
FALSE
DISPLAY
OUTPUT
STOP

Example

- **5 x 3 [0101 x 0011]**

Let us consider,
M = bin(5) where N=4 : M=0101
Q = bin (3) where N=4: Q=0011
Let us consider the output to be R
R = AQ where,
A = 0000, Q = 0011 where q0=1, q1=1, q2=0, q3=0 and q-1=0(initial)

Iteration 1:
4 >0 , Comparing q0 and q-1 => 1 0
.’. A = A – M (A=A+2’M) => A = 0000+1011 = 1011
Output before ASR=10110011: After arithmetic shift right = 11011001 -> (LSB removed=q-1=1,current LSB=1=q0)
N-1=3

Iteration 2:
3 > 0 therefore compare q0 and q-1: 11
.’. Output before ASR=11011001: After arithmetic shift right =11101100-> (LSB removed=q-1=1,current LSB=0=q0)
N-1=2

Iteration 3:
2 > 0 therefore compare q0 and q-1: 0 1
.’. A = A + M = 1110+0101=0011
.’. Output before ASR=00111100: After arithmetic shift right =00011110-> (LSB removed=q-1=0,current LSB=0=q0)
N-1=1

Iteration 4:
1 > 0 therefore compare q0 and q-1: 0 0
.’. Output before ASR=00011110: After arithmetic shift right =00001111-> (LSB removed=q-1=1,current LSB=0=q0)
N-1=0 N !> 0 [Loop terminates]

Hence required output = 00001111 = 15

Source Code

```
AREA PROGRAM, CODE, READONLY
MOV R2, #3 ; Load Q into R2
MOV R3, #0 ; Load A into R3
MOV R4, #4 ; Number of iterations
MOV R5, #5 ; Load M into R5

LOOP

CMP R4, #0
BLE END_LOOP
AND R0, R2, #1
LSR R1, R2, #1
AND R1, R1, #1
ORR R1, R1, R0
CMP R1, #2
BEQ SUB_M

SUB_M
SUB R0, R3, #5
MOV R3, R0
B SHIFT_RIGHT ; Performing right shift
CMP R1, #1
BEQ ADD_M

ADD_M
ADD R0, R3, #5
MOV R3, R0
SHIFT_RIGHT

SHIFT_RIGHT
ASR R3, R3, #1
ROR R2, R2,#1
SUBS R4, R4, #1
B ITERATION_LOOP
ORR R3, R3,R0

STOP B STOP
END
```

Conclusion

Booth's algorithm transforms digital arithmetic by optimizing the multiplication of signed binary numbers. Through its nuanced analysis of binary patterns, it minimizes computational steps, enhancing efficiency. Its applicability extends to hardware design and various computational domains, where its streamlined approach proves invaluable. Booth's algorithm represents a hallmark of innovation and effectiveness in digital computation, shaping the landscape of modern technology with its efficient and versatile methodology.

Thanks!