

# DAYANANDA SAGAR UNIVERSITY

Devarakaggalahalli, Harohalli Kanakapura Road, Dt, Ramanagara, Karnataka 562112



**SCHOOL OF  
ENGINEERING**

**Bachelor of Technology in  
COMPUTER SCIENCE AND ENGINEERING  
(Artificial Intelligence and Machine Learning)**



**COURSE TITLE: COMPUTER ORGANISATION AND ARCHITECTURE**

**ASSIGNMENT: 1**

**“Booth Algorithm implementation”**

By

**Akhila Rao D - ENG22AM0002.**

**Gaana Shree S - ENG22AM0014.**

**Gayatri Govinda Setty - ENG22AM0017.**

**Himashree L - ENG22AM0025.**

**Under the supervision of**

**Prof. Pradeep Kumar K**

Assistant Professor, Artificial Intelligence & Machine Learning, SOE

**Dr. Vegi Fernando**

Associate Professor, Artificial Intelligence & Machine Learning, SOE

**Dr. Joshuva Arockia Dhanraj**

Associate Professor, Artificial Intelligence & Machine Learning, SOE

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**(Artificial Intelligence and Machine Learning)**

**SCHOOL OF ENGINEERING**

**DAYANANDA SAGAR UNIVERSITY, BANGALORE**





# SCHOOL OF ENGINEERING

School of Engineering  
Department of Computer Science & Engineering  
(Artificial Intelligence and Machine Learning)  
Devarakaggalahalli, Harohalli Kanakapura Road, Dt, Ramanagara, Karnataka 562112



## CERTIFICATE

This is to certify that the Mini-Project titled "Booth's multiplication algorithm" is carried out by Akhila Rao D (ENG22AM0002), Gaana Shree S (ENG22AM0014), Gayatri Govinda Setty (ENG22AM0017) and Himashree L (ENG22AM0025) bonafide students of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence and Machine Learning) at the School of Engineering, Dayananda Sagar University,

*[Signature]*  
21/5/2024

Prof. Pradeep Kumar K  
Assistant Professor.  
Dept. of CSE(AI&ML),  
School of Engineering  
Dayananda Sagar University

*[Signature]*  
Dr Jayavinda Vrindavanam  
Head of Department  
CSE (AI-ML)  
School of Engineering  
Dayananda Sagar University

*[Signature]*  
29/5/24

Dr Vegi Fernando.  
Associate Professor  
Dept. of CSE(AI&ML),  
School of Engineering  
Dayananda Sagar University

*[Signature]*  
22/5/24  
Dr. Joshua Arockia Dhanraj  
Associate Professor  
Dept. of CSE(AI&ML),  
School of Engineering  
Dayananda Sagar University



## ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr.Udaya Kumar Reddy K R , Dean, School of Engineering, Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Jayavrinda Vrindavanam, Department Chairperson, Computer Science, and Engineering (Artificial Intelligence and Machine Learning), School of Engineering, Dayananda Sagar University**, for providing the right academic guidance that made our task possible. We would like to thank our guide **Prof. Pradeep Kumar K, Assistant Professor, Dept. of Computer Science and Engineering(Artificial Intelligence and Machine Learning), Dr. Vegi Fernando Prof. Joshuva Arockia Dhanraj, Associate Professor, Dept. of Computer Science and Engineering(Artificial Intelligence and Machine Learning), School of Engineering, Dayananda Sagar University**, for sparing their valuable time to extend help in every step of our UG Research project work, which paved the way for smooth progress and the fruitful culmination of the research.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Research work.

## TABLE OF CONTENTS

S.No	Topic	Page No
1	Introduction	1
2	Problem statement	2
3	Objectives	2
4	Algorithm	3-4
5	Flowchart	5
6	Example	6-7
7	Source code	8
8	Result and analysis	9
9	Reference	10



## INTRODUCTION:

In the world of computers, multiplication is super important. In real life, we use multiplication for things like calculating prices when shopping, determining areas and volumes in construction, understanding growth rates in finance and economics, analyzing data in science and research, and much more. 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. So, whether you're using math in everyday life or in high-tech fields, multiplication plays a key role in making things work smoothly and efficiently. So, finding a faster and simpler way to multiply numbers is a big deal in making computers work better. 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.

## PROBLEM STATEMENT:

Design an algorithmic technique that minimizes both time complexity and the number of steps, particularly focusing on reducing shift operations, for multiplying large integers, including both signed and unsigned multiplication.

## OBJECTIVES:

- **Efficient Multiplication:** Design an algorithm that minimizes the number of addition operations required to compute the product of two binary numbers.
- **Handling Signed Numbers:** Ensure correct handling of signed binary numbers, as Booth's algorithm can be adapted for both signed and unsigned multiplication.
- **Versatility:** Create a solution adaptable for implementation in both software and hardware environments, optimizing for speed, memory usage, and versatility.
- **Accuracy:** Ensure accurate computation of the product, meeting the precision requirements of the application.
- **Performance Analysis:** Evaluate the performance of Booth's algorithm in comparison to traditional multiplication methods, considering factors such as execution time, memory usage, and scalability.



## ALGORITHM:

### STEP 1: Initialization

- **START:** The algorithm begins here.
- **OUTPUT = A, Q(0) Q(-1):** The initial values of the accumulator (A) and the two least significant bits (Q(0) and Q(-1)) of the quotient register (Q) are possibly shown here, although these values are typically set to 0 in the initialization step. The quotient register is used to hold the multiplier during the algorithm.
- **N= NO. OF BIT:** A counter variable named Count is set to n, where n is the number of bits in the multiplier.

### STEP 2: Iterate through Multiplier Bits

- **10:** The flowchart checks the two least significant bits (Q(0) and Q(-1)) of register Q. If these bits are equal to 10 (binary 2), it signifies encountering the first 1 in a sequence of 1s in the multiplier. **A-A-M:** In this case, the multiplicand (M) needs to be subtracted from the accumulator (A). This accounts for the sequence of 1s preceding the 0.
- **01:** If the two bits are equal to 01 (binary 1), it indicates the end of a sequence of 1s and the beginning of a sequence of 0s. **A+M:** The multiplicand (M) is added to the accumulator (A). This is because the 01 pattern signifies a change from positive to negative in the contribution from the multiplier bits.
- **00/11:** No operation is performed on A if the two bits are either 11 (binary 3) or 00 (binary 0). These patterns (consecutive 1s or 0s) don't introduce any change to the partial product.

### STEP 3: Shift and Count

- **1. ASR(OUTPUT):** The contents of registers holding the output (A), Q(0), and Q(-1) are shifted right by one bit position using an Arithmetic Shift Right (ASR) operation. This effectively divides the partial product in A by 2 while maintaining the sign extension in the most significant bit.

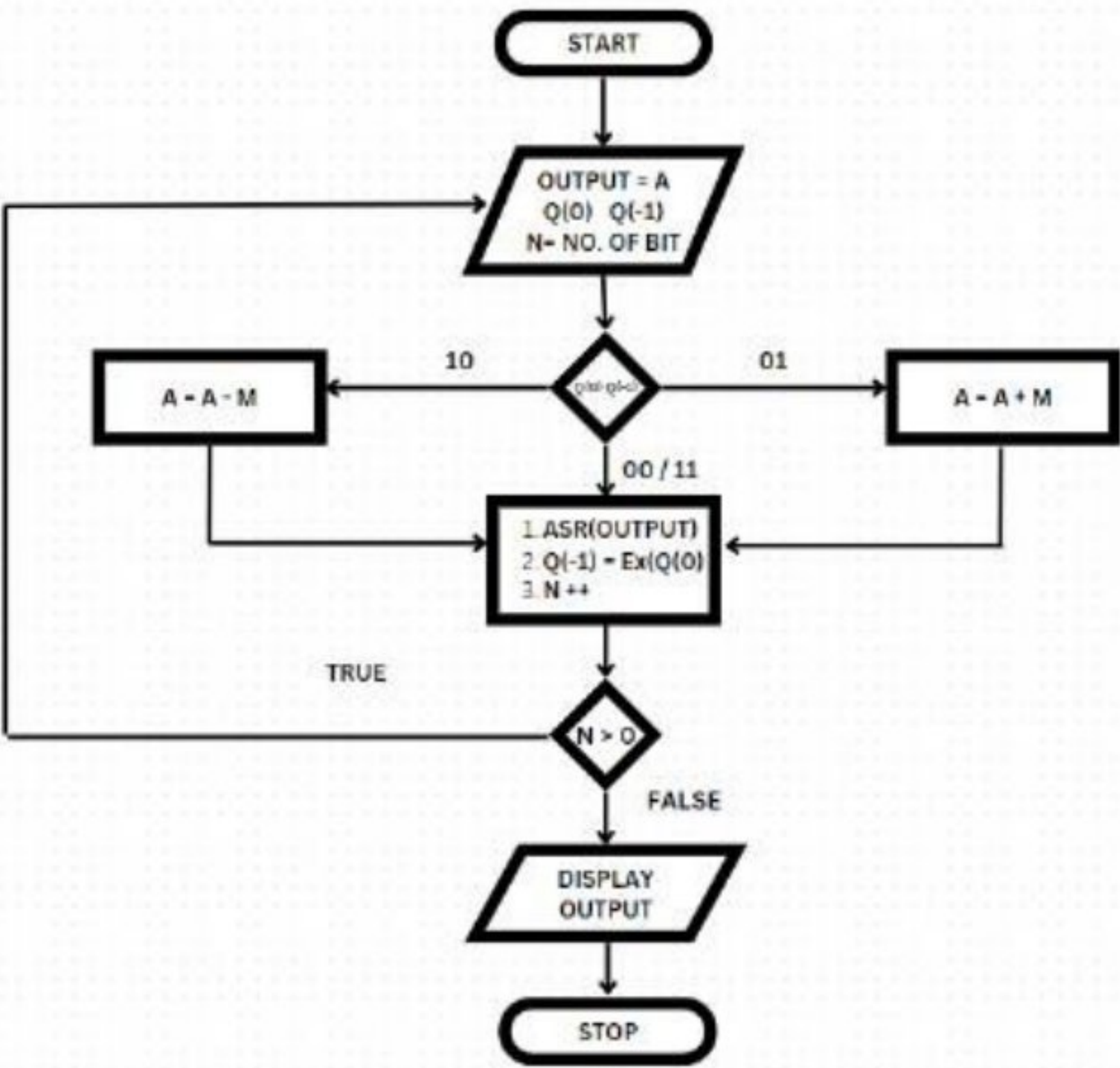
- **2.  $Q(-1) = Ex(Q(0))$  :** The sign bit of the output ( $Q(-1)$ ) is set to the previous value of the least significant bit ( $Q(0)$ ) of the quotient register. This is done to prepare for examining bit pairs in the next iteration.

#### **STEP 4: Loop Termination**

- **$N > 0$ :** The flowchart checks if the counter ( $N$ ) is still greater than 0. If  $N$  is zero, it signifies that all bits in the multiplier have been processed.
  - **TRUE:** If  $N$  is greater than zero, the loop continues back to step 2 to process the next bit pair of the multiplier.
  - **FALSE:** If  $N$  is not greater than zero, the algorithm terminates. The final value in register A holds the signed product of the multiplicand ( $M$ ) and the multiplier ( $Q$ ).



FLOWCHART:





## EXAMPLE: $5 \times 3$ [0101 x 0011]

Let us consider,

$M = \text{bin}(5)$  where  $N=4$  :  $M=0101$

$Q = \text{bin}(3)$  where  $N=4$ :  $Q=0011$

Let us consider the output to be  $R$

$R = AQ$  where,

$A = 0000$ ,  $Q = 0011$  where  $q_0=1$ ,  $q_1=1$ ,  $q_2=0$ ,  $q_3=0$  and  $q_{-1}=0$ (initial)

### Iteration 1:

$4 > 0$  , Comparing  $q_0$  and  $q_{-1} \Rightarrow 1\ 0$

$\therefore A = A - M$  ( $A=A+2^N M$ )  $\Rightarrow A = 0000+1011 = 1011$

Output before ASR=10110011: After arithmetic shift right = 11011001  $\rightarrow$  (LSB removed= $q_{-1}=1$ , current LSB= $1=q_0$ )

$N-1=3$

### Iteration 2:

$3 > 0$  therefore compare  $q_0$  and  $q_{-1}$ : 11

$\therefore$  Output before ASR=11011001: After arithmetic shift right =11101100 $\rightarrow$  (LSB removed= $q_{-1}=1$ , current LSB= $0=q_0$ )

$N-1=2$

### Iteration 3:

$2 > 0$  therefore compare  $q_0$  and  $q_{-1}$ : 0 1

$\therefore A = A + M = 1110+0101=0011$

$\therefore$  Output before ASR=00111100: After arithmetic shift right =00011110 $\rightarrow$  (LSB removed= $q_{-1}=0$ , current LSB= $0=q_0$ )

$N-1=1$



#### Iteration 4:

$1 > 0$  therefore compare  $q_0$  and  $q_{-1}$ : 0 0

∴ Output before ASR=00011110: After arithmetic shift right =00001111-> (LSB removed= $q_{-1}=1$ , current LSB=0= $q_0$ )

$N-1=0$   $N \neq 0$  [ Loop terminates]

Hence required output = **00001111 = 15**

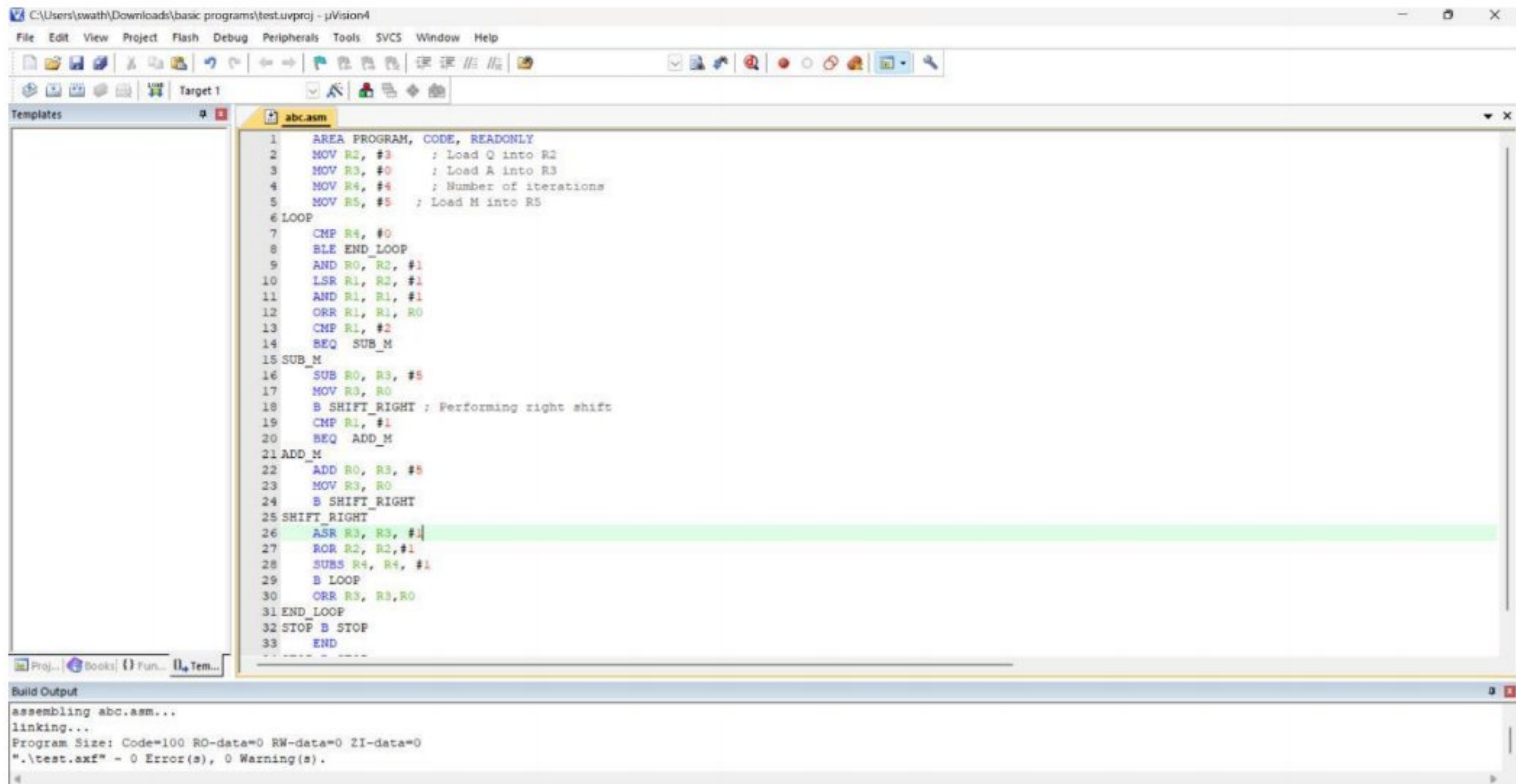


## SOURCE CODE:

```
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
```



# RESULT AND ANALYSIS:

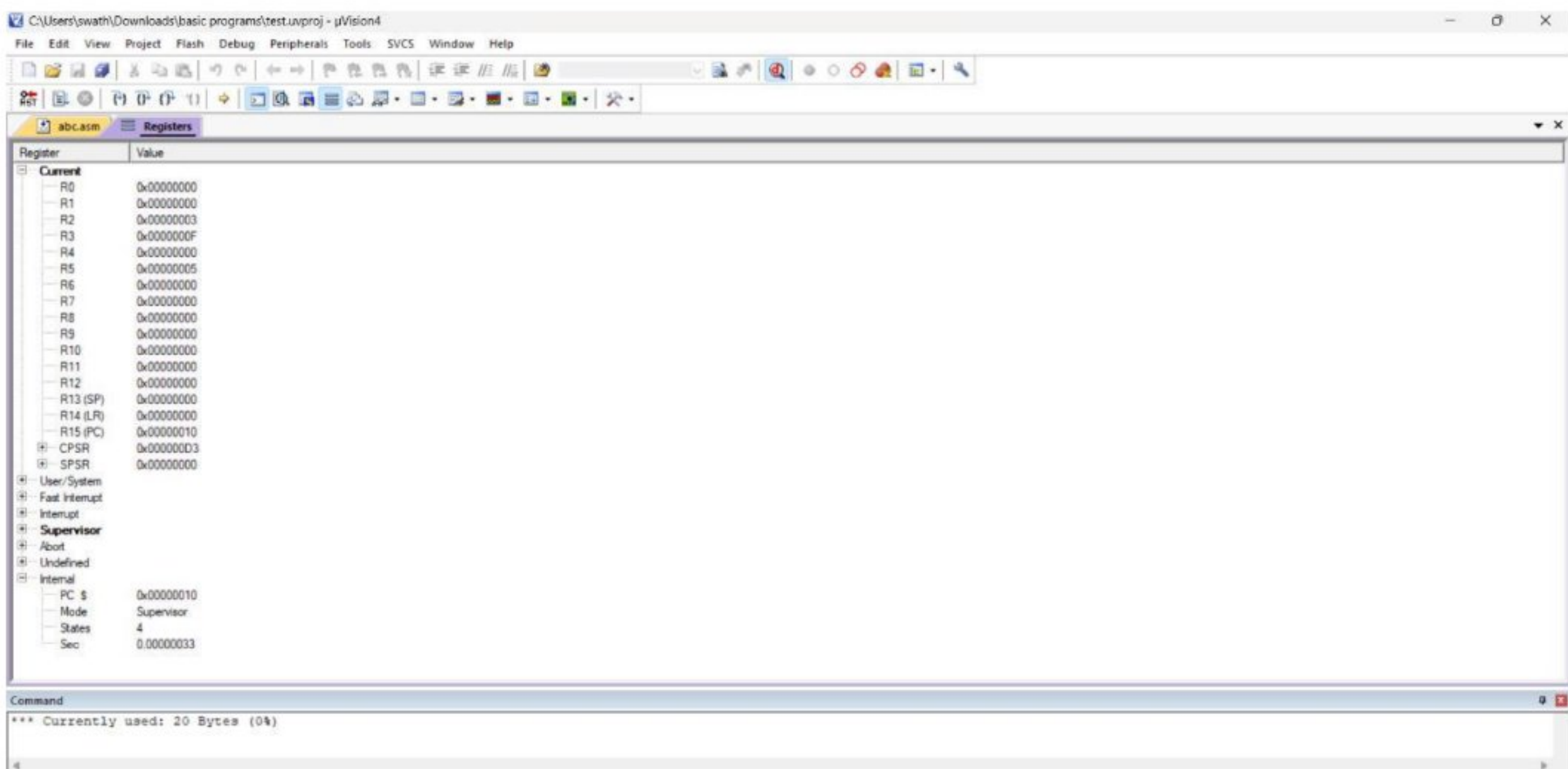


The screenshot shows the uVision4 IDE with the assembly file 'abc.asm' open. The code is as follows:

```
1 AREA PROGRAM, CODE, READONLY
2 MOV R2, #3 ; Load Q into R2
3 MOV R3, #0 ; Load A into R3
4 MOV R4, #4 ; Number of iterations
5 MOV R5, #5 ; Load M into R5
6 LOOP
7 CMP R4, #0
8 BLE END_LOOP
9 AND R0, R2, #1
10 LSR R1, R2, #1
11 AND R1, R1, #1
12 ORR R1, R1, R0
13 CMP R1, #2
14 BEQ SUB_M
15 SUB_M
16 SUB R0, R3, #5
17 MOV R3, R0
18 B SHIFT_RIGHT ; Performing right shift
19 CMP R1, #1
20 BEQ ADD_M
21 ADD_M
22 ADD R0, R3, #5
23 MOV R3, R0
24 B SHIFT_RIGHT
25 SHIFT_RIGHT
26 ASR R3, R3, #1
27 ROR R2, R2, #1
28 SUBS R4, R4, #1
29 B LOOP
30 ORR R3, R3, R0
31 END_LOOP
32 STOP B STOP
33 END
```

The 'Build Output' window at the bottom shows the following messages:

```
assembling abc.asm...
linking...
Program Size: Code=100 RO-data=0 RW-data=0 ZI-data=0
".\test.axf" - 0 Error(s), 0 Warning(s).
```



The screenshot shows the uVision4 IDE with the Register window open. The registers and their values are as follows:

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000003
R3	0x0000000F
R4	0x00000000
R5	0x00000005
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC	0x00000010
Mode	Supervisor
States	4
Sec	0.00000033

The Command window at the bottom shows the following message:

```
*** Currently used: 20 Bytes (0%)
```



## REFERENCES:

1. <https://medium.com/@jetnipit54/booth-algorithm-e6b8a6c5b8d>
2. <https://www.geeksforgeeks.org/computer-organization-booths-algorithm/>
3. <https://www.javatpoint.com/booths-multiplication-algorithm-in-coa>