



Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

Experiment No. 9
Implement Non-Restoring algorithm using c-programming
Name: Pratik Mankar
Roll No: 30
Date of Performance: 19/09/24
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Aim - To implement Non-Restoring division algorithm using c-programming.

Objective -

1. To understand the working of Non-Restoring division algorithm.
2. To understand how to implement Non-Restoring division algorithm using c-programming.

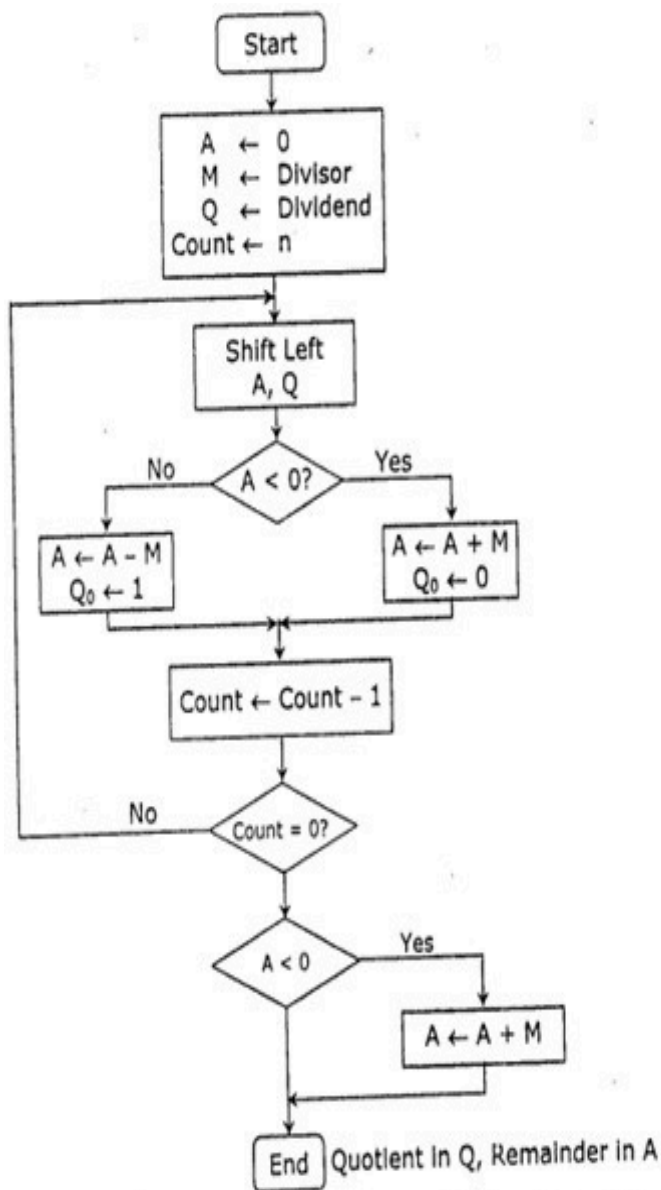
Theory:

In each cycle content of the register, A is first shifted and then the divisor is added or subtracted with the content of register A depending upon the sign of A. In this, there is no need of restoring, but if the remainder is negative then there is a need of restoring the remainder. This is the faster algorithm of division.



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science



Perform $8 \div 3$ by non-restoring division technique.

	A Register	Q Register	
Initially	0 0 0 0 0	1 0 0 0	
Shift	0 0 0 0 1	0 0 0 □	
Subtract	1 1 1 0 1		
Set Q_0	① 1 1 1 0	0 0 0 ①	First Cycle
Shift	1 1 1 0 0	0 0 ① □	
Add	0 0 0 1 1		
Set Q_0	① 1 1 1 1	0 0 ① ①	Second Cycle
Shift	1 1 1 1 0	0 ① ① □	
Add	0 0 0 1 1		
Set Q_0	① 0 0 0 1	0 0 ① ①	Third Cycle
Shift	0 0 0 1 0	0 ① ① □	
Subtract	1 1 1 0 1		
Set Q_0	① 1 1 1 1	0 0 ① ①	Fourth Cycle
Add	1 1 1 1 1		Quotient
	0 0 0 1 1		
	0 0 0 1 0		Remainder



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Program -

```
#include <stdio.h>
```

```
void binaryPrint(int n, int bits) {  
    for (int i = bits - 1; i >= 0; i--) {  
        printf("%d", (n >> i) & 1);  
    }  
    printf("\n");  
}
```

```
int main() {  
    int M, Q, A = 0, count;  
    int n;  
  
    printf("Enter the divisor (M): ");  
    scanf("%d", &M);  
    printf("Enter the dividend (Q): ");  
    scanf("%d", &Q);  
    printf("Enter the number of bits: ");  
    scanf("%d", &n);
```

```
    count = n;
```

```
    printf("\nInitial values:\n");  
    printf("A: ");  
    binaryPrint(A, n);  
    printf("Q: ");  
    binaryPrint(Q, n);  
    printf("M: ");  
    binaryPrint(M, n);  
    printf("\n");
```

```
    while (count > 0) {  
        A = (A << 1) | ((Q >> (n - 1)) & 1);  
        Q = Q << 1;
```

```
        printf("After left shift:\n");  
        printf("A: ");  
        binaryPrint(A, n);
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
printf("Q: ");
binaryPrint(Q, n);

if (A >= 0) {
    A = A - M;
    printf("After subtraction (A >= 0):\n");
}else {
    A = A + M;
    printf("After addition (A < 0):\n");
}

printf("A: ");
binaryPrint(A, n);

if (A >= 0) {
    Q = Q | 1;
}else {
    Q = Q & ~(1);
}

printf("After updating Q0:\n");
printf("A: ");
binaryPrint(A, n);
printf("Q: ");
binaryPrint(Q, n);
printf("\n");

count--;
}

if (A < 0) {
    A = A + M;
    printf("Final correction (if A < 0, add M to A):\n");
    printf("A: ");
    binaryPrint(A, n);
}

printf("\nFinal quotient (Q): ");
binaryPrint(Q, n);
printf("Final remainder (A): ");
binaryPrint(A, n);
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
return 0;  
}
```

Output:

```
Enter the divisor (M): 4  
Enter the dividend (Q): 2  
Enter the number of bits: 4  
  
Initial values:  
A: 0000  
Q: 0010  
M: 0100  
  
After left shift:  
A: 0000  
Q: 0100  
After subtraction (A >= 0):  
A: 1100  
After updating Q0:  
A: 1100  
Q: 0100  
  
After left shift:  
A: 1000  
Q: 1000  
After addition (A < 0):  
A: 1100  
After updating Q0:  
A: 1100  
Q: 1000  
  
After left shift:  
A: 1001  
Q: 0000  
After addition (A < 0):  
A: 1101  
After updating Q0:  
A: 1101  
Q: 0000
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
Final correction (if A < 0, add M to A):  
A: 0010
```

```
Final quotient (Q): 0000  
Final remainder (A): 0010
```

Conclusion -

In this experiment, we successfully implemented the Non-Restoring Division Algorithm in C to divide two unsigned integers represented in binary form. The algorithm effectively demonstrates the process of binary arithmetic, including addition, subtraction, and bitwise shifting. Through step-by-step execution, we observed how the quotient and remainder are derived based on the initial dividend and divisor. This implementation not only reinforces the understanding of binary operations but also highlights the efficiency of Non-Restoring Division in handling division tasks without requiring restoration in every step. Overall, the experiment provides valuable insights into algorithm design and binary number manipulation in programming.