Experiment No. 8
Implement Restoring algorithm using c-programming
Name: James Lewis
Roll Number: 28
Date of Performance:19-09-2024
Date of Submission:

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

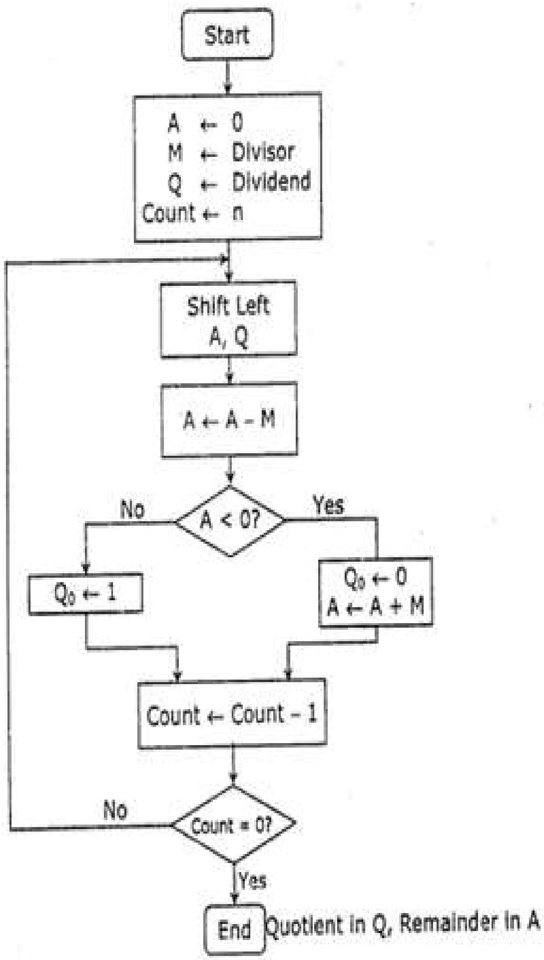
Objective -

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

Theory:

1) The divisor is placed in M register, the dividend placed in Q register.
2) At every step, the A and Q registers together are shifted to the left by 1-bit
3) M is subtracted from A to determine whether A divides the partial remainder. If it does, then Q0 set to 1-bit. Otherwise, Q0 gets a 0 bit and M must be added back to A to restore the previous value.
4) The count is then decremented and the process continues for n steps. At the end, the quotient is in the Q register and the remainder is in the A register.

Flowchart



Perform 8 ÷ 3 by restoring division technique.

	A Register	Q Register	
Initially	0 0 0 0 0	1 0 0 0	First Cycle
Shift	0 0 0 0 1	0 0 0 □	
Subtract M	1 1 1 0 1		
Set Q_0	1 1 1 1 0		
Restore(A+M)	0 0 0 1 1		
	0 0 0 0 1	0 0 0 0	
Shift	0 0 0 1 0	0 0 0 □	Second Cycle
Subtract M	1 1 1 0 1		
Set Q_0	1 1 1 1 1		
Restore(A+M)	0 0 0 1 1		
	0 0 0 1 0	0 0 0 0	
Shift	0 0 1 0 0	0 0 0 □	Third Cycle
Subtract M	1 1 1 0 1		
Set Q_0	0 0 0 0 1		
		0 0 0 1	
Shift	0 0 0 1 0	0 0 1 □	Fourth Cycle
Subtract M	1 1 1 0 1		
Set Q_0	1 1 1 1 1		
Restore(A+M)	0 0 0 1 1		
	0 0 0 1 0	0 0 1 0	
	Remainder	Quotient	

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

```
        A = A - M;

        printf("After subtraction:\n");
        printf("A: ");
        binaryPrint(A, n);

        if (A < 0) {
            A = A + M;
            Q = Q & ~(1);
        } else {
            Q = Q | 1;
        }

        printf("After restore (if needed):\n");
        printf("A: ");
        binaryPrint(A, n);
        printf("Q: ");
        binaryPrint(Q, n);
        printf("\n");

        count--;
    }

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

    return 0;
}
```

Output -

```
Enter the divisor (M): 8
Enter the dividend (Q): 3
Enter the number of bits: 4

Initial values:
A: 0000
Q: 0011
M: 1000
```

```
After left shift:
A: 0000
Q: 0110
After subtraction:
A: 1000
After restore (if needed):
A: 0000
Q: 0110

After left shift:
A: 0000
Q: 1100
After subtraction:
A: 1000
After restore (if needed):
A: 0000
Q: 1100

After left shift:
A: 0001
Q: 1000
After subtraction:
A: 1001
After restore (if needed):
A: 0001
Q: 1000

After left shift:
A: 0011
Q: 0000
After subtraction:
A: 1011
After restore (if needed):
A: 0011
Q: 0000

Final quotient (Q): 0000
Final remainder (A): 0011
```

```
Enter the divisor (M): 8
Enter the dividend (Q): 3
Enter the number of bits: 4

Initial values:
A: 0000
Q: 0011
M: 1000

After left shift:
A: 0000
Q: 0110
After subtraction:
A: 1000
After restore (if needed):
A: 0000
Q: 0110

After left shift:
A: 0000
Q: 1100
After subtraction:
A: 1000
After restore (if needed):
A: 0000
Q: 1100

After left shift:
A: 0001
Q: 1000
After subtraction:
A: 1001
After restore (if needed):
A: 0001
Q: 1000

After left shift:
A: 0011
Q: 0000
After subtraction:
A: 1011
After restore (if needed):
A: 0011
Q: 0000

Final quotient (Q): 0000
Final remainder (A): 0011
```

Conclusion -

- The Restoring Division Algorithm successfully divides two integers by performing repeated subtraction and shifting operations.
- After the completion of n steps, where n is the number of bits in the divisor, the quotient is found in the Q register and the remainder in the A register.

This C program demonstrates how the Restoring Division Algorithm works by simulating the process shown in your flowchart.