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

Experiment No. 9
Implement Non-restoring algorithm using c-programming
Name: Patil Pranjal Keshav
Roll Number: 45
Date of Performance:
Date of Submission:

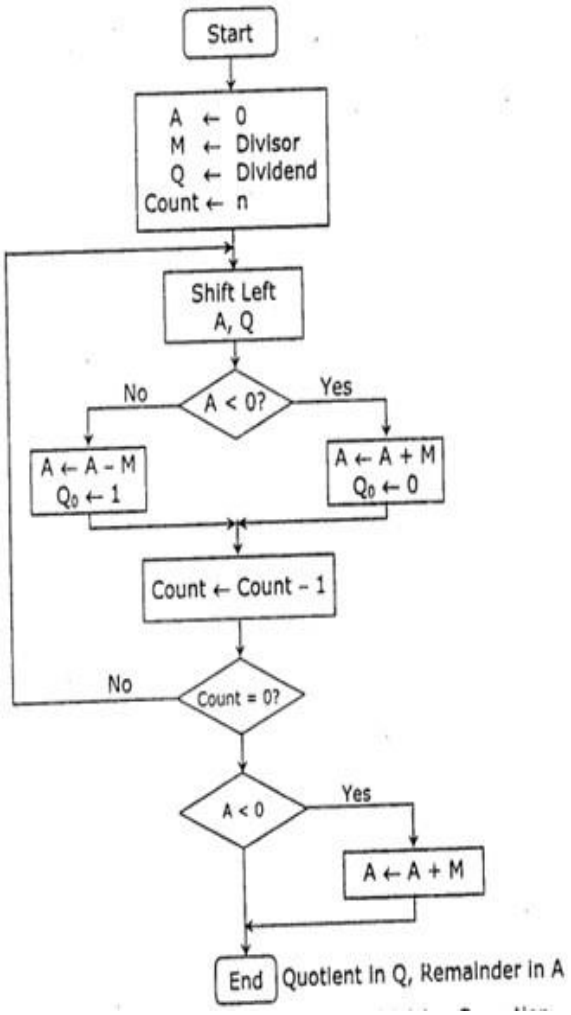
Aim - To implement Non-Restoring division algorithm using c-programming.

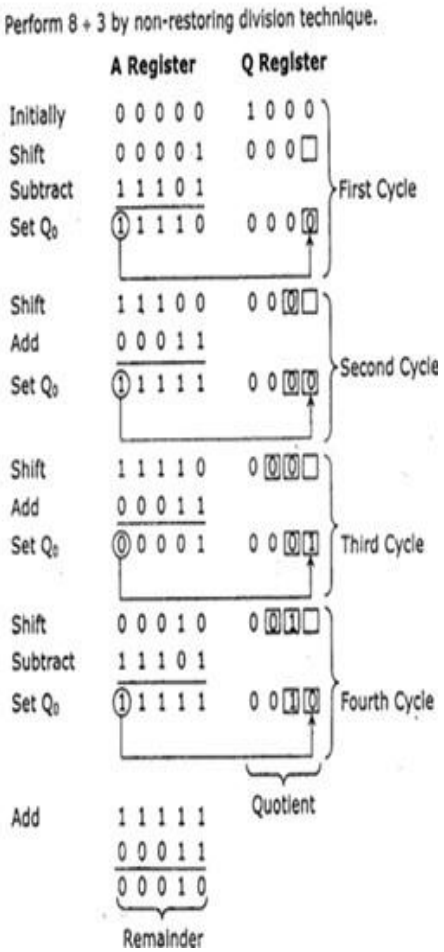
Objective -

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

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.





Program –

```
#include <stdio.h>

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

void nonRestoringDivision(int dividend, int divisor) {
   int A = 0; // Accumulator
   int Q = dividend; // Dividend
   int M = divisor; // Divisor
   int n = 4; // Number of bits (adjust as needed)
   int Q0 = 0; // Previous bit of Q

   printf("A\tQ\tComments\n");
   printf("0000\t");
   binaryPrint(Q, n);
   printf("\tStart\n");

   // Iterate for the number of bits in the dividend
   for (int i = 0; i < n; i++) {
      // Left shift A and Q
      A = (A << 1) | ((Q >> (n - 1)) & 1);
      Q = (Q << 1);
      printf("%04d\t", A);
      binaryPrint(Q, n);
      printf("\tLeft Shift A,Q\n");

      // A = A - M
      A = A - M;
      if (A < 0) {
         // If A < 0, set Qo to 0
         Q0 = 0;
         A = A + M; // Restore A
      } else {
         // If A >= 0, set Qo to 1
         Q0 = 1;
      }
```

```
// Update Q with Qo
        Q |= Q0;
        printf("%04d\t", A);
        binaryPrint(Q, n);
        printf("\tA = A - M; Q0=%d; A = A + M\n", Q0);

        // Left shift A and Q again
        A = (A << 1) | Q0; // Prepare for next step
        printf("%04d\t", A);
        binaryPrint(Q, n);
        printf("\tLeft Shift A,Q\n");
    }

    printf("Quotient = ");
    binaryPrint(Q, n);
    printf("\nRemainder = ");
    binaryPrint(A, n);
    printf("\n");
}

int main() {
    int dividend, divisor;

    // Input from user
    printf("Enter the Dividend: ");
    scanf("%d", &dividend);
    printf("Enter the Divisor: ");
    scanf("%d", &divisor);

    nonRestoringDivision(dividend, divisor);

    return 0;
}
```

Output:

```
Enter the Dividend: 10
Enter the Divisor: 2
A       Q       Comments
0000    1010    Start
0001    0100    Left Shift A,Q
1111    0100    A = A - M; Q0=0; A = A + M
0010    1000    Left Shift A,Q
0000    1000    A = A - M; Q0=1; A = A + M
0001    0000    Left Shift A,Q
Quotient = 0101
Remainder = 0000
```

Conclusion -

This program effectively simulates the Non-Restoring Division algorithm, demonstrating each step in the process. The final quotient and remainder are displayed in binary format, illustrating how the algorithm arrives at the result. If you have any questions or need further assistance, feel free to ask!