Batch: D3 Roll No.:16010123294
Experiment / assignment / tutorial No. 04

TITLE : To study and implement Non Restoring method of division

AIM : The basis of algorithm is based on paper and pencil approach and the operation involve repetitive shifting with addition and subtraction. So the main aim is to depict the usual process in the form of an algorithm.

Expected OUTCOME of Experiment: (Mention CO/CO's attained here)

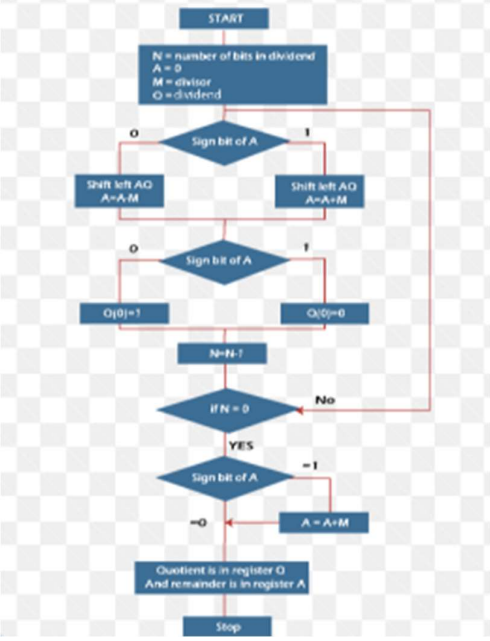
Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.
3. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization", First Edition, Wiley-India.

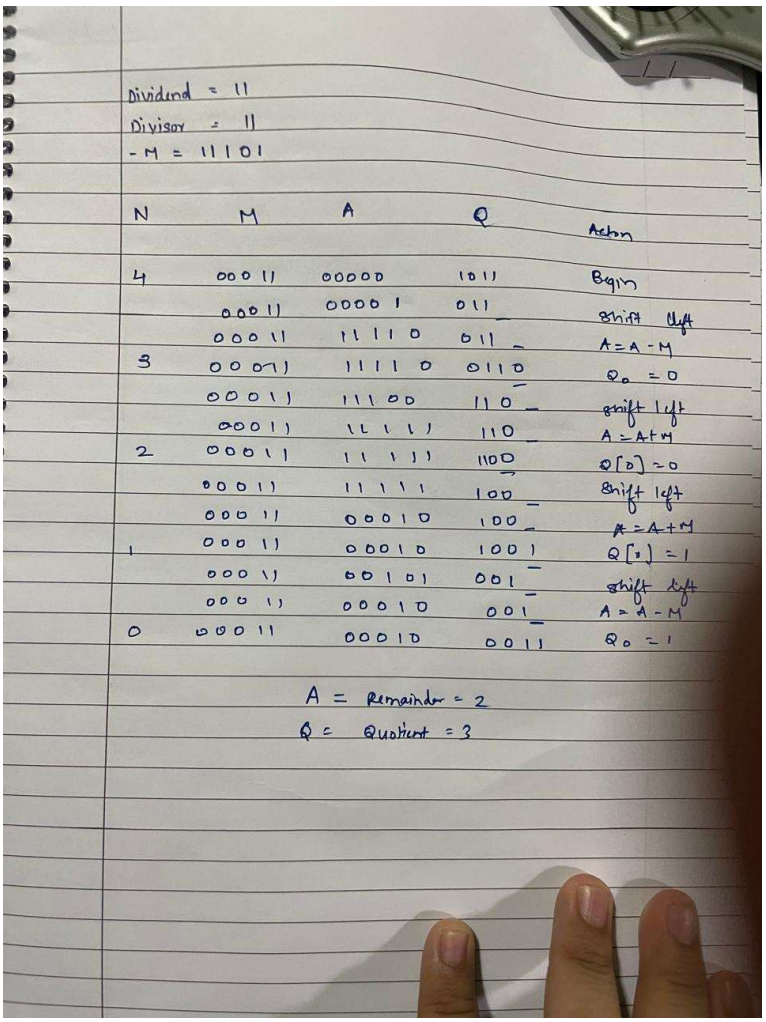
Pre Lab/ Prior Concepts:

The Non Restoring algorithm works with any combination of positive and negative numbers.

Flowchart for Non Restoring of Division(Students need to draw)



Example: (Handwritten solved problem needs to uploaded)



Implementation:

```
import java.util.Scanner;

public class NonRestoringDivision {

    public static String add(String A, String M) {
        int carry = 0;
        StringBuilder sum = new StringBuilder();

        for (int i = A.length() - 1; i >= 0; i--) {
```

```
        int temp = (A.charAt(i) - '0') + (M.charAt(i) - '0') + carry;
        if (temp > 1) {
            sum.append(temp % 2);
            carry = 1;
        } else {
            sum.append(temp);
            carry = 0;
        }
    }

    return sum.reverse().toString();
}

public static String complement(String m) {
    StringBuilder M = new StringBuilder();

    for (int i = 0; i < m.length(); i++) {
        M.append((m.charAt(i) - '0' + 1) % 2);
    }

    M = new StringBuilder(add(M.toString(), "0001"));
    return M.toString();
}

public static void nonRestoringDivision(String Q, String M, String A) {
    int count = M.length();
    String compM = complement(M);
    boolean successful = true;

    System.out.println("Initial Values: A: " + A + " Q: " + Q + " M: " + M);

    while (count > 0) {
        System.out.print("\nStep: " + (M.length() - count + 1) + " Left Shift and ");

        A = A.substring(1) + Q.charAt(0);

        if (successful) {
            A = add(A, compM);
            System.out.println("Subtract:");
        } else {
            A = add(A, M);
            System.out.println("Addition:");
        }

        System.out.print("A: " + A + " Q: " + Q.substring(1) + "_");
```

```
            if (A.charAt(0) == '1') {
                Q = Q.substring(1) + "0";
                System.out.println("  -Unsuccessful");
                successful = false;
                System.out.println("A: " + A + " Q: " + Q + " -Addition in next Step");
            } else {
                Q = Q.substring(1) + "1";
                System.out.println("  Successful");
                successful = true;
                System.out.println("A: " + A + " Q: " + Q + " -Subtraction in next step");
            }

            count--;
        }

        System.out.println("\nQuotient(Q): " + Q + " Remainder(A): " + A);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter dividend (binary): ");
        String dividend = scanner.next();

        System.out.print("Enter divisor (binary): ");
        String divisor = scanner.next();

        String accumulator = "0".repeat(dividend.length());

        nonRestoringDivision(dividend, divisor, accumulator);

        scanner.close();
    }
}
```

Output:

```
Enter dividend (binary): 1010
Enter divisor (binary): 0010
Initial Values: A: 0000 Q: 1010 M: 0010

Step: 1 Left Shift and Subtract:
A: 1111 Q: 010_  -Unsuccessful
A: 1111 Q: 0100 -Addition in next Step

Step: 2 Left Shift and Addition:
A: 0000 Q: 100_  Successful
A: 0000 Q: 1001 -Subtraction in next step

Step: 3 Left Shift and Subtract:
A: 1111 Q: 001_  -Unsuccessful
A: 1111 Q: 0010 -Addition in next Step

Step: 4 Left Shift and Addition:
A: 0000 Q: 010_  Successful
A: 0000 Q: 0101 -Subtraction in next step

Quotient(Q): 0101 Remainder(A): 0000
```

Conclusion

Applied knowledge of java programming to code algorithm of non restoring division

Post Lab Descriptive Questions

What are the advantages of non restoring division over restoring division?

Non-restoring division is faster and simpler than restoring division because it eliminates the need for an additional correction step after each subtraction, which reduces the number of operations and hardware complexity.

Date: _____