Batch: E-2 **Roll No.: 16010123325**

Experiment / assignment / tutorial No. 4

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)

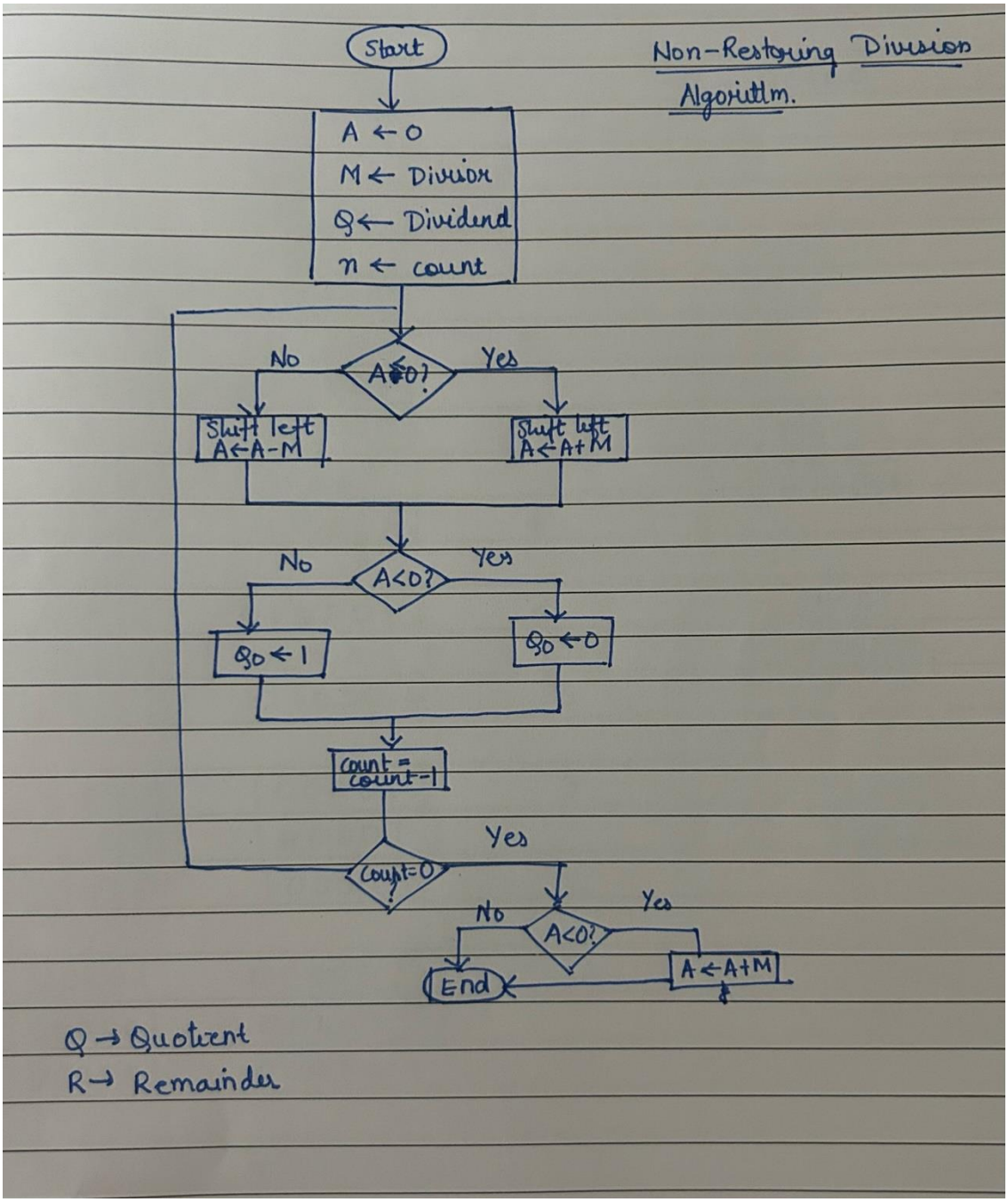
Books/ Journals/ Websites referred:

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.

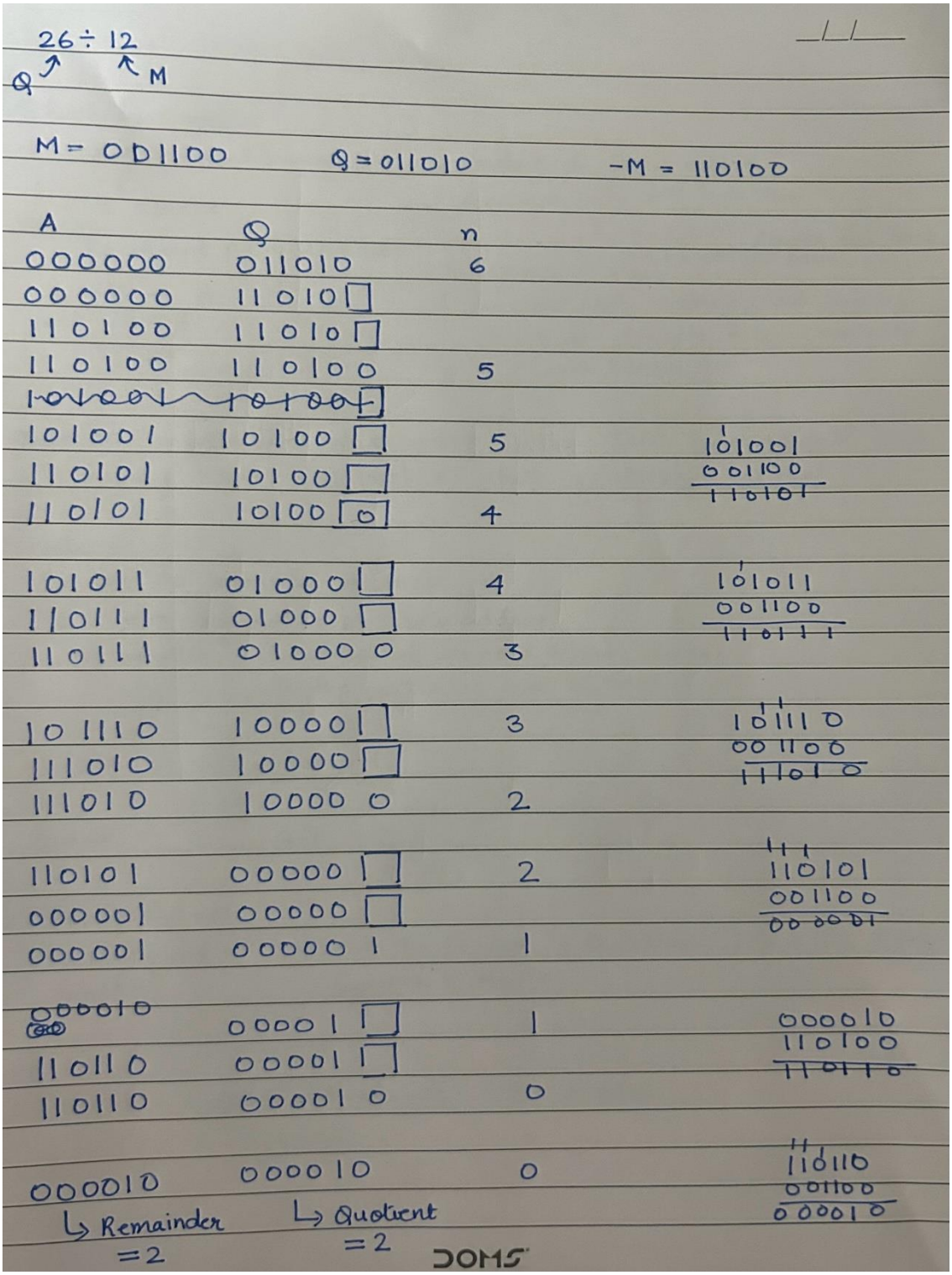
Pre Lab/ Prior Concepts:

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)



Code:

```
#include <bits/stdc++.h>
using namespace std;
#define uint unsigned int

void nonRestoringDivision(uint dividend, uint divisor, uint *quotient,
uint* remainder) {
    uint acc = 0;
    uint q = dividend;
    uint n = sizeof(uint) * 8;

    for (int i = 0; i < n; ++i) {

        acc = (acc << 1) | ((q >> (n-1)) & 1);
        q <<= 1;

        acc -= divisor;

        if (acc & (1 << (n-1))) {
            acc += divisor;
        } else {
            q |= 1;
        }
    }

    *quotient = q;
    *remainder = acc;
}


int main() {
    uint dividend, divisor;
    uint quotient, remainder;
    cout << "Enter Dividend: ";
    cin >> dividend;
    cout << "Enter Divisor: ";
    cin >> divisor;

    if (divisor == 0) {
        cerr << "Error: Division by Zero.\n";
        return 1;
    }
```

```
    nonRestoringDivision(dividend, divisor, &quotient, &remainder);
    cout << "Quotient: " << quotient << '\n';
    cout << "Remainder: " << remainder << '\n';
}
```

Output:

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs\" ; if
 ($?) { g++ non-restoring-division.cpp -o non-restoring-division } ; if ($?) { .\non-restoring-division }
Enter Dividend: 26
Enter Divisor: 12
Quotient: 2
Remainder: 2
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs> cd "c:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Progra
ms\" ; if ($?) { g++ non-restoring-division.cpp -o non-restoring-division } ; if ($?) { .\non-restoring-division }
Enter Dividend: 12
Enter Divisor: 3
Quotient: 4
Remainder: 0
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

Conclusion

The above experiment highlights the implementation of Non-Restoring Division algorithm using C++.

Post Lab Descriptive Questions

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

Non-restoring division offers several advantages over restoring division:

1. **Speed**: Non-restoring division generally results in faster computation. Unlike restoring division, it eliminates the need for additional steps to restore the original value after a failed subtraction, reducing the overall number of operations.

2. **Simplified hardware**: Non-restoring division requires less hardware complexity since it avoids the back-and-forth adjustment of the remainder in restoring division.

Date: _____