Batch: E2	**Roll No.:** 16010123325
Experiment / assignment / tutorial No._____	

TITLE : To study and implement Restoring method of division

AIM : The basis of algorithm is based on paper and pencil approach and the operation involves 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)

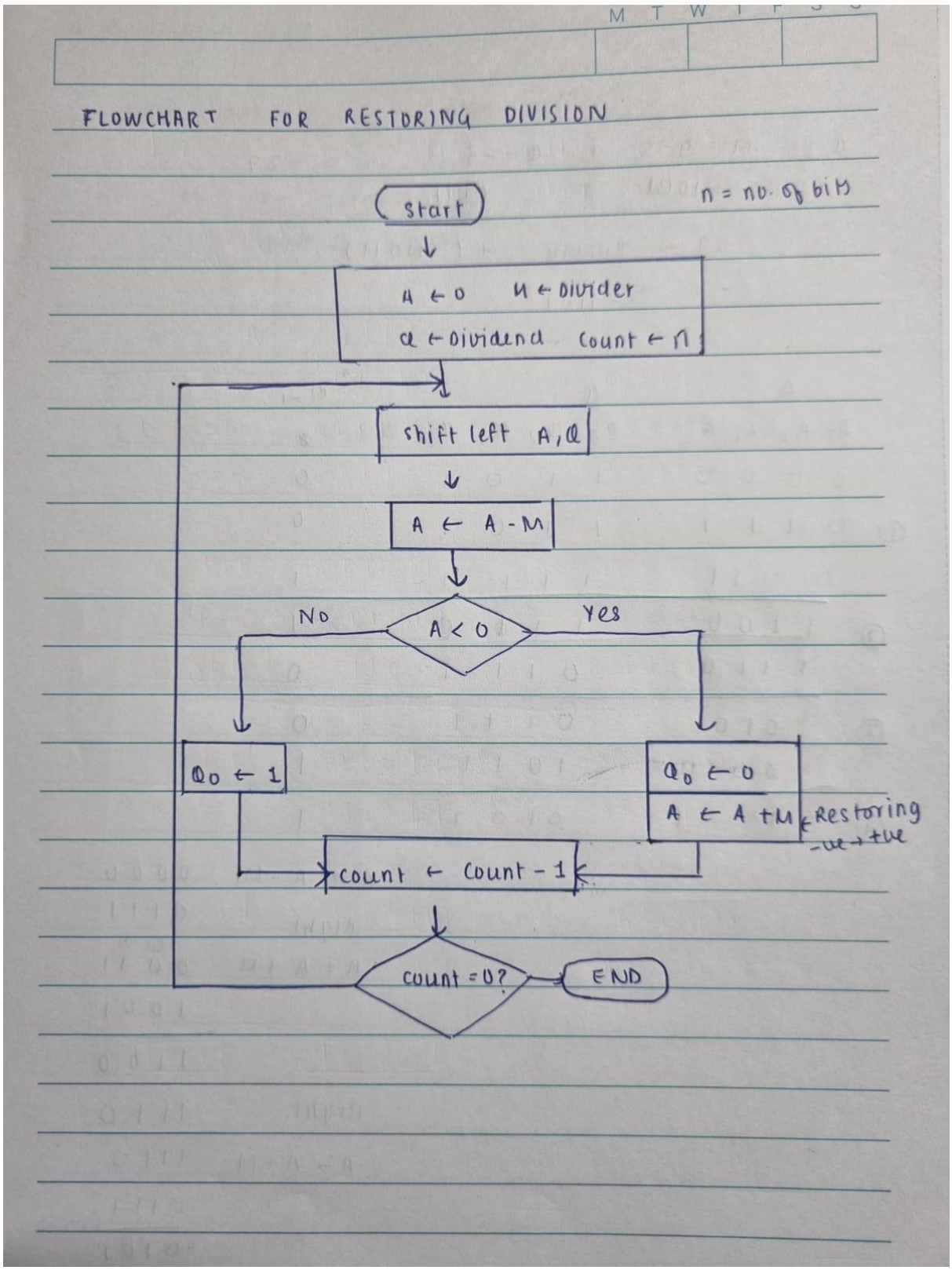
Books/ Journals/ Websites referred:

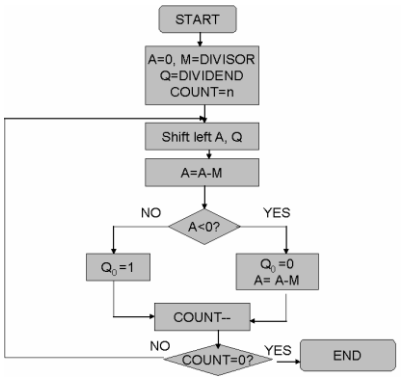
1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.
3. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization",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

Pre Lab/ Prior Concepts:

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

Flowchart for Restoring of Division:

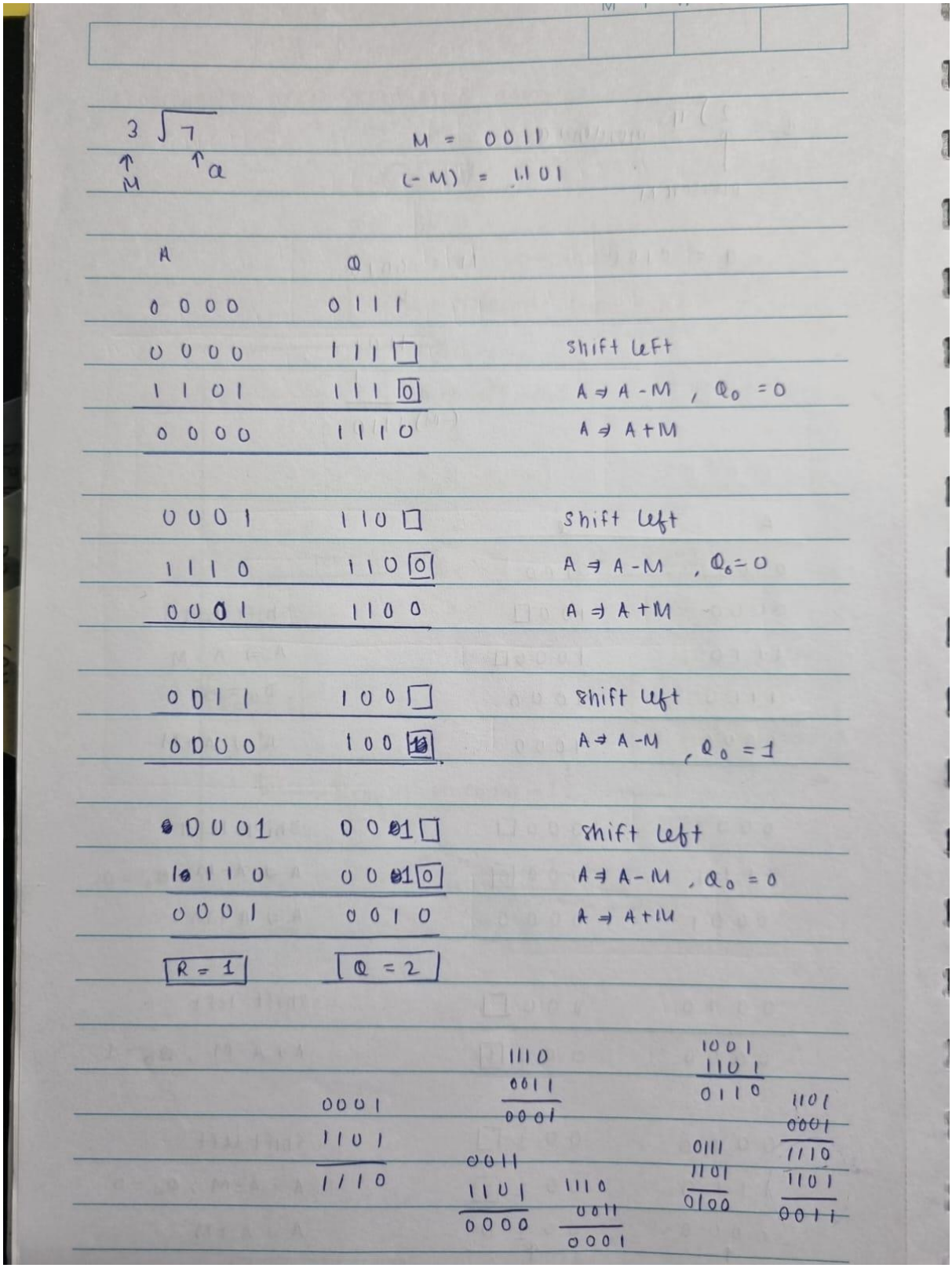




Design Steps:

1. Start
2. Initialize A=0, M=Divisor, Q=Dividend and count=n (no of bits)
3. Left shift A, Q
4. If MSB of A and M are same
5. Then A=A-M
6. Else A=A+M
7. If MSB of previous A and present A are same
8. Q_0=0 & store present A
9. Else Q_0=0 & restore previous A
10. Decrement count.
11. If count=0 go to 11
12. Else go to 3
13. STOP

Example:- (Handwritten solved problems needs to be uploaded)



Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

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

    *quotient = 0;

    for (int i = 0; i < n; ++i) {
        acc = (acc << 1 | (q >> (n - 1) & 1));
        q <<= 1;

        acc -= divisor;

        if (acc < 0) {
            acc += divisor;
        } else {
            q |= 1;
        }
    }

    *quotient = q;
    *remainder = acc;
}

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

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

    restoringDivision(dividend, divisor, &quotient, &remainder);

    cout << "Quotient: " << quotient << '\n';
    cout << "Remainder: " << remainder << '\n';
```

```
    return 0;
}
```

```
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA> cd "c:\Users\Shrey\OneDrive\Desk
top\KJSCE\SEM-3\COA\Programs\" ; if ($?) { g++ restoring-division.cpp -o restoring-d
ivision } ; if ($?) { .\restoring-division }
Enter dividend: 7
Enter divisor: 3
Quotient: 2
Remainder: 1
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\OneD
rive\Desktop\KJSCE\SEM-3\COA\Programs\" ; if ($?) { g++ restoring-division.cpp -o re
storing-division } ; if ($?) { .\restoring-division }
Enter dividend: 10
Enter divisor: 5
Quotient: 2
Remainder: 0
PS C:\Users\Shrey\OneDrive\Desktop\KJSCE\SEM-3\COA\Programs>
```

Conclusion:

The Restoring method of division can be implemented in C++ to efficiently calculate the quotient of two integers by iteratively subtracting the divisor from the dividend and restoring the remainder.

Post Lab Descriptive Questions

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

Restoring division and non-restoring division are two different approaches to performing division in digital arithmetic. Here are the advantages of restoring division over non-restoring division:

Advantages of Restoring Division:

1. **Simpler Implementation**: Restoring division is easier to implement in hardware, as it only requires a simple subtract-and-shift operation in each iteration. This simplicity makes it more suitable for implementation in digital circuits.

2. **Faster Convergence**: Restoring division converges faster than non-restoring division, especially for larger dividend values. This is because restoring division corrects the remainder in each iteration, ensuring that the quotient is accurate.

3. **Easier Error Detection**: In restoring division, errors can be easily detected by checking the remainder at each iteration. If the remainder is not zero, an error has occurred. This makes error detection and correction more straightforward.
4. **Better Suitability for Signed Division**: Restoring division is more suitable for signed division (i.e., dividing two signed numbers), as it can handle the sign of the result correctly.

Date: _____