Name: Ashwera Hasan
Experiment 4
Batch: B2 **Roll: 16010124107**

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)

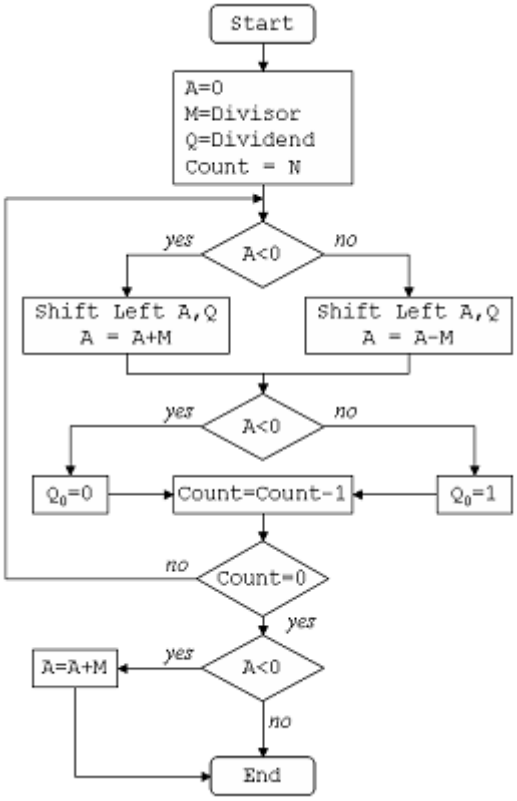
Books/ Journals/ Websites referred:

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.

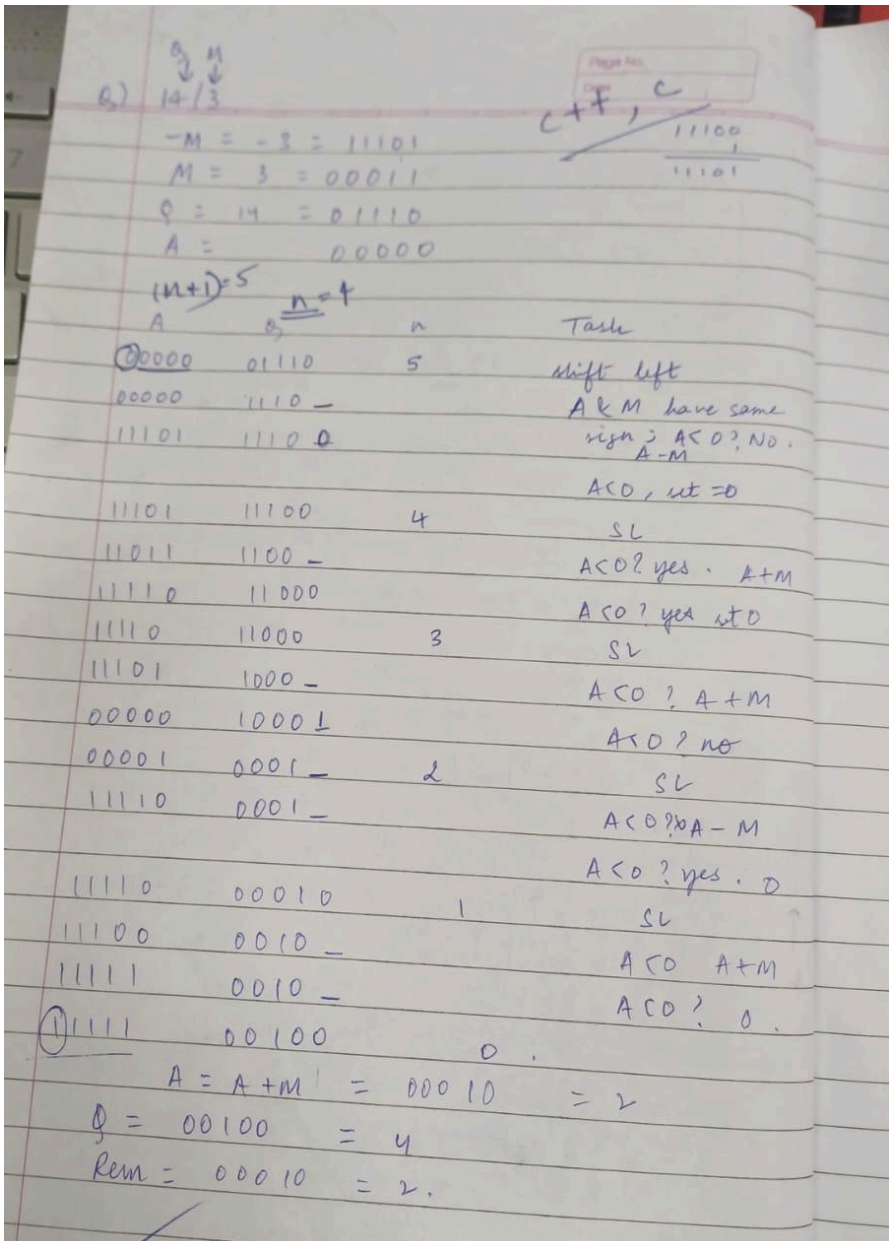
Flowchart for Non Restoring of Division(Students need to draw):-



Algorithm:

1. Start
2. Set A=0, M=divisor, and Q = dividend. Convert to binary. Set N=number of bits+1. Set count=n+1
3. Run this till count=0.
4. Shift A and Q to the left
5. If MSB of A is 1, do A= A+M
6. else, do A-M = A
7. after this operation, if MSB of A is 1, set the last bit of Q=1 else set 0
8. Reduce count.
9. Repeat steps 3 till 8.
10. If finally after count=0, MSB of A is 1, perform A = A+M
11. Quotient is in Q and Remainder is in A.
12. Stop.

Example: (Handwritten solved problem needs to uploaded):-

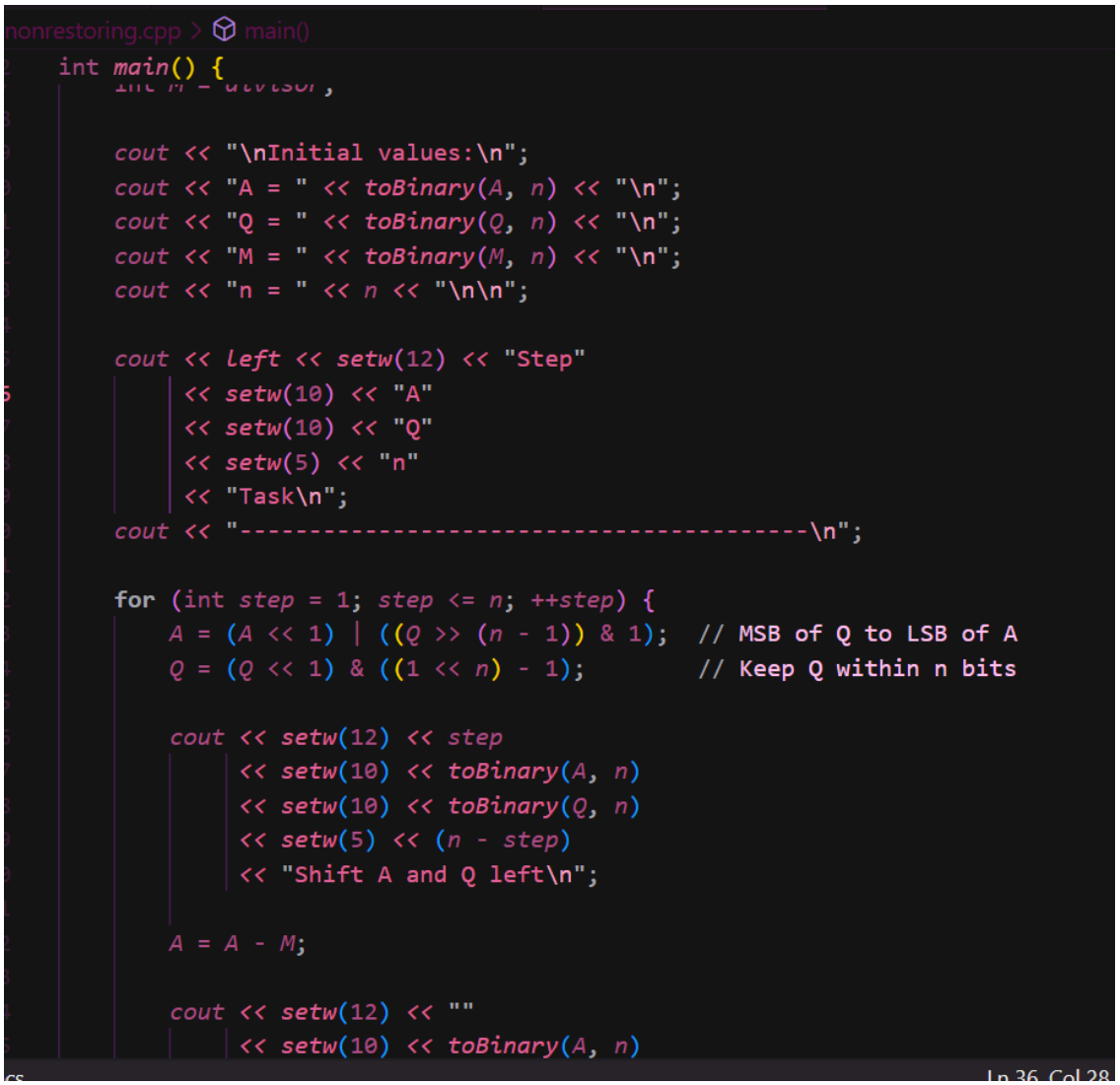

Q) 14/3
-M = -3 = 11101
M = 3 = 00011
Q = 14 = 01110
A = 00000
(n+1)=5
n=4
A Q n Task
00000 01110 5 shift left
00000 1110_ A & M have same sign; A<0? No. A-M
11101 1110 0 A<0, set =0
11101 11100 4 SL
11011 1100_ A<0? yes. A+M
11110 11000 A<0? yes set 0
11110 11000 3 SL
11101 1000_ A<0? A+M
00000 1000 1 A<0? no
00001 0001_ 2 SL
11110 0001_ A<0? no A-M
11110 00010 1 A<0? yes. 0
11100 0010_ SL
11111 0010_ A<0 A+M
11111 00100 0 A<0? 0.
A = A+M = 00010 = 2
Q = 00100 = 4
Rem = 00010 = 2.

CODE:-

nonrestoring.cpp

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

string toBinary(int num, int bits) {
    string bin = "";
    for (int i = bits - 1; i >= 0; --i)
        bin += ((num >> i) & 1) ? '1' : '0';
    return bin;
}

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

    if (dividend < 0 || divisor <= 0) {
        cout << "Only positive integers supported.\n";
        return 1;
    }

    int n = 4;
    int A = 0;
    int Q = dividend;
    int M = divisor;

    cout << "\nInitial values:\n";
```

0 secs


nonrestoring.cpp > main()
int main() {
cout << "\nInitial values:\n";
cout << "A = " << toBinary(A, n) << "\n";
cout << "Q = " << toBinary(Q, n) << "\n";
cout << "M = " << toBinary(M, n) << "\n";
cout << "n = " << n << "\n\n";
cout << left << setw(12) << "Step"
<< setw(10) << "A"
<< setw(10) << "Q"
<< setw(5) << "n"
<< "Task\n";
cout << "--\n";
for (int step = 1; step <= n; ++step) {
A = (A << 1) | ((Q >> (n - 1)) & 1); // MSB of Q to LSB of A
Q = (Q << 1) & ((1 << n) - 1); // Keep Q within n bits
cout << setw(12) << step
<< setw(10) << toBinary(A, n)
<< setw(10) << toBinary(Q, n)
<< setw(5) << (n - step)
<< "Shift A and Q left\n";
A = A - M;
cout << setw(12) << ""
<< setw(10) << toBinary(A, n)

cc.cpp | submissionisallyouneed.cpp | nonrestoring.cpp

nonrestoring.cpp > main()

```cpp
int main() {
    for (int step = 1; step <= n; ++step) {

        cout << setw(12) << ""
             << setw(10) << toBinary(A, n)
             << setw(10) << toBinary(Q, n)
             << setw(5) << ""
             << "A = A - M\n";

        if (A < 0) {
            A = A + M;
            Q = Q & (~1);
            cout << setw(12) << ""
                 << setw(10) << toBinary(A, n)
                 << setw(10) << toBinary(Q, n)
                 << setw(5) << ""
                 << "A < 0, Restore A, Q0 = 0\n";
        } else {
            Q = Q | 1;
            cout << setw(12) << ""
                 << setw(10) << toBinary(A, n)
                 << setw(10) << toBinary(Q, n)
                 << setw(5) << ""
                 << "A >= 0, Q0 = 1\n";
        }
    }

    cout << "\nFinal Quotient (Q): " << toBinary(Q, n) << " (" << Q << ")\n";
    cout << "Final Remainder (A): " << toBinary(A, n) << " (" << A << ")\n";
```

OUTPUT:-

```
 g++ nonrestoring.cpp -o nonrestoring } ; if ($?) { .\nonrestori
Enter Dividend: 14
Enter Divisor: 3

Initial values:
A = 0000
Q = 1110
M = 0011
n = 4

Step        A         Q         n    Task
-----------------------------------------
1           0001      1100      3    Shift A and Q left
            1110      1100           A = A - M
            0001      1100           A < 0, Restore A, Q0 = 0
2           0011      1000      2    Shift A and Q left
            0000      1000           A = A - M
            0000      1001           A >= 0, Q0 = 1
3           0001      0010      1    Shift A and Q left
            1110      0010           A = A - M
            0001      0010           A < 0, Restore A, Q0 = 0
4           0010      0100      0    Shift A and Q left
            1111      0100           A = A - M
            0010      0100           A < 0, Restore A, Q0 = 0

Final Quotient (Q): 0100 (4)
Final Remainder (A): 0010 (2)
```

Conclusion:-

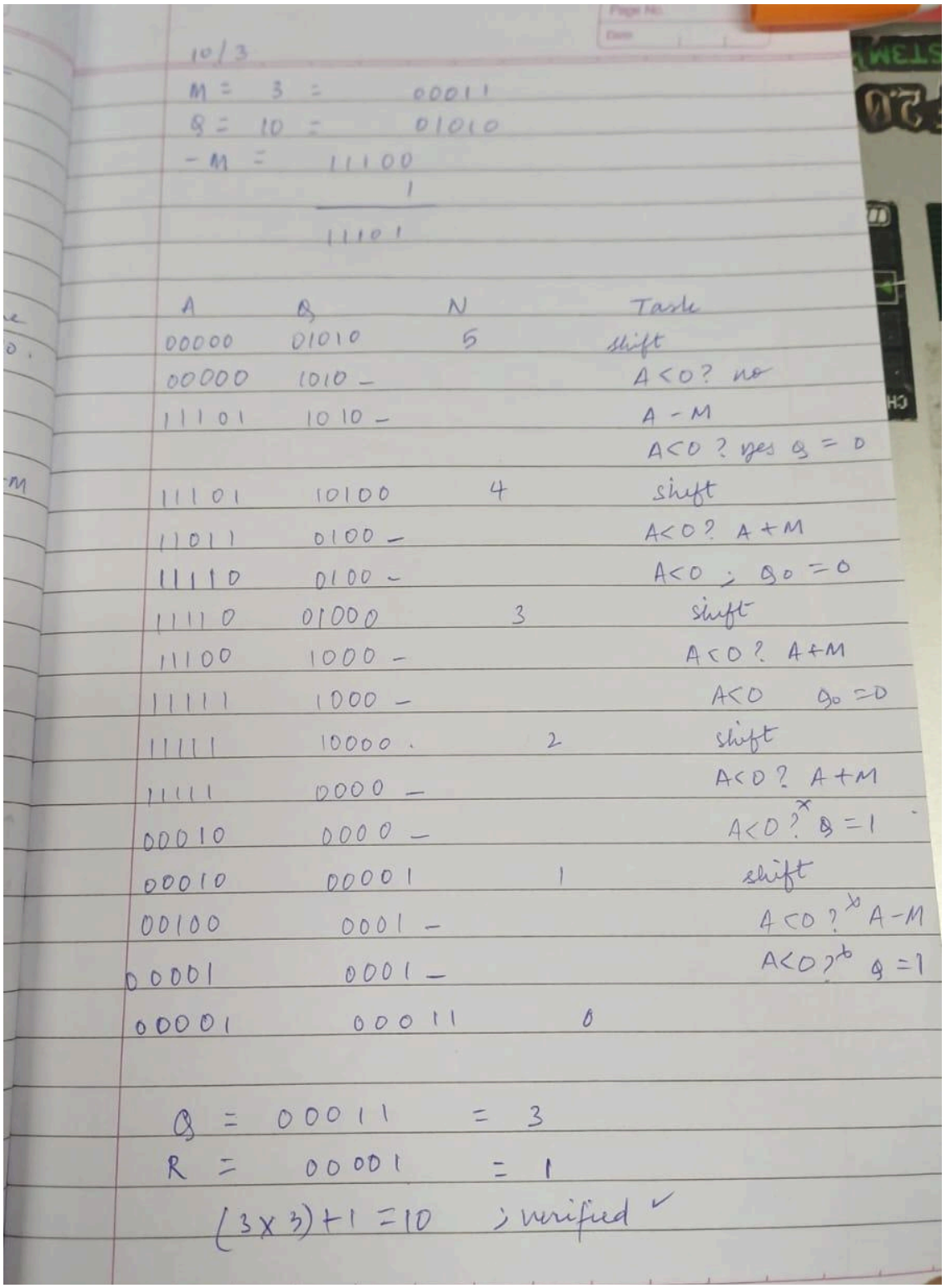
The non-restoring division is used for unsigned binary values that simplifies the procedure by eliminating the restoring phase. The non-restoring division is simpler and more effective than restoring division. It just employs addition and subtraction operations instead of restoring division, which requires extra steps to restore the original result after a failed subtraction.

Post Lab Descriptive Questions

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

1. Non restoring division is faster and simpler
2. It is more effective
3. It avoids extra steps of restoring that used to obtain the previous A after a failed subtraction.
4. Requires less hardware in implementation.

Q. Solve 10/3 using Non-Restoring algorithm for division operation?


10/3
M = 3 = 00011
Q = 10 = 01010
-M = 11100
1
11101
A Q N Task
00000 01010 5 shift
00000 1010 – A<0? no
11101 1010 – A-M
A<0? yes q=0
11101 10100 4 shift
11011 0100 – A<0? A+M
11110 0100 – A<0; q0=0
11110 01000 3 shift
11100 1000 – A<0? A+M
11111 1000 – A<0 q0=0
11111 10000 2 shift
11111 0000 – A<0? A+M
00010 0000 – A<0? q=1
00010 00001 1 shift
00100 0001 – A<0? A-M
00001 0001 – A<0? q=1
00001 00011 0
Q = 00011 = 3
R = 00001 = 1
(3x3)+1 = 10 ; verified

Date:- 08.08.25