Batch: A3	**Roll No.: 16010121045**
Experiment / assignment / tutorial No: 3	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

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)

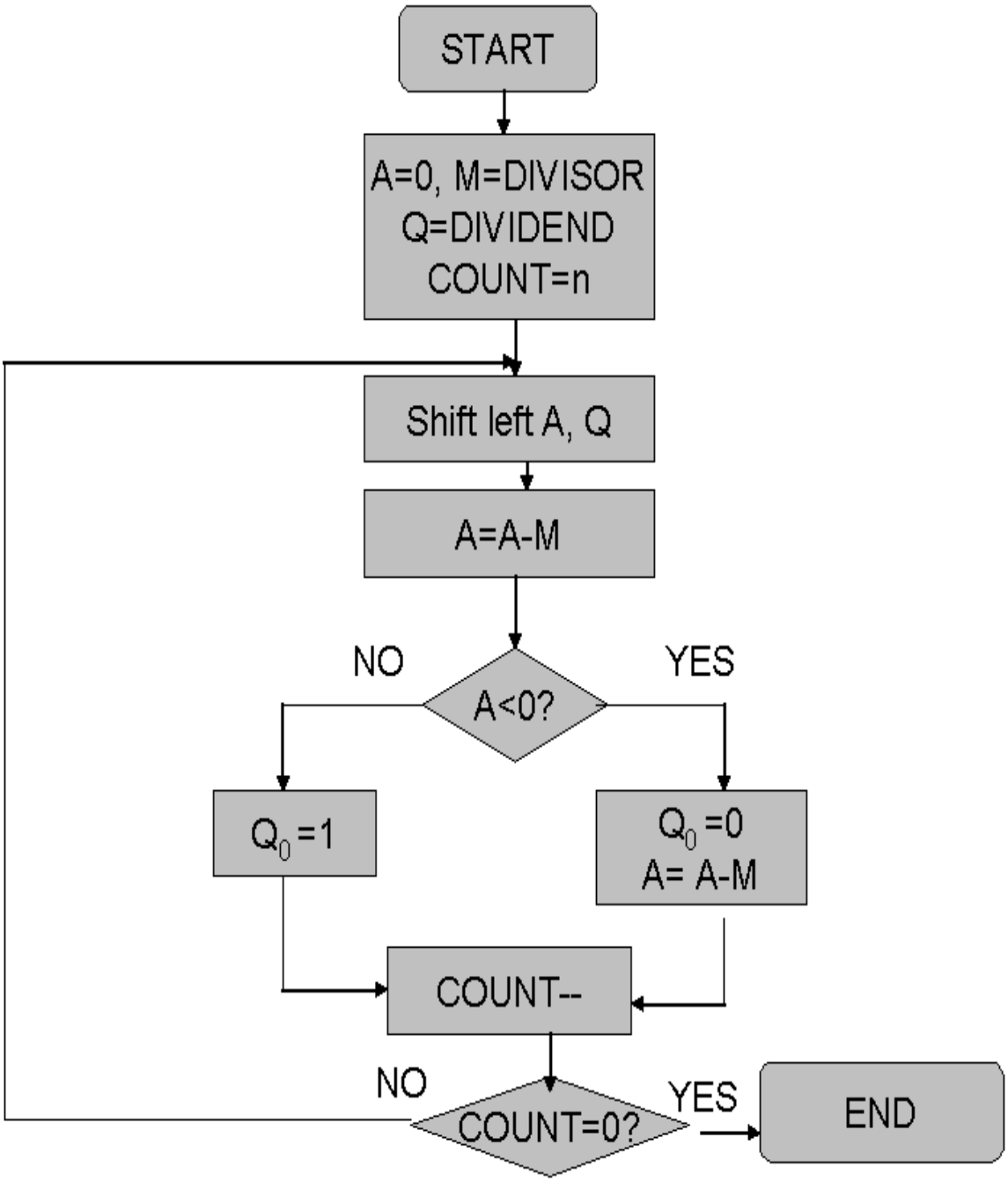
Books/ Journals/ Websites referred:

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.

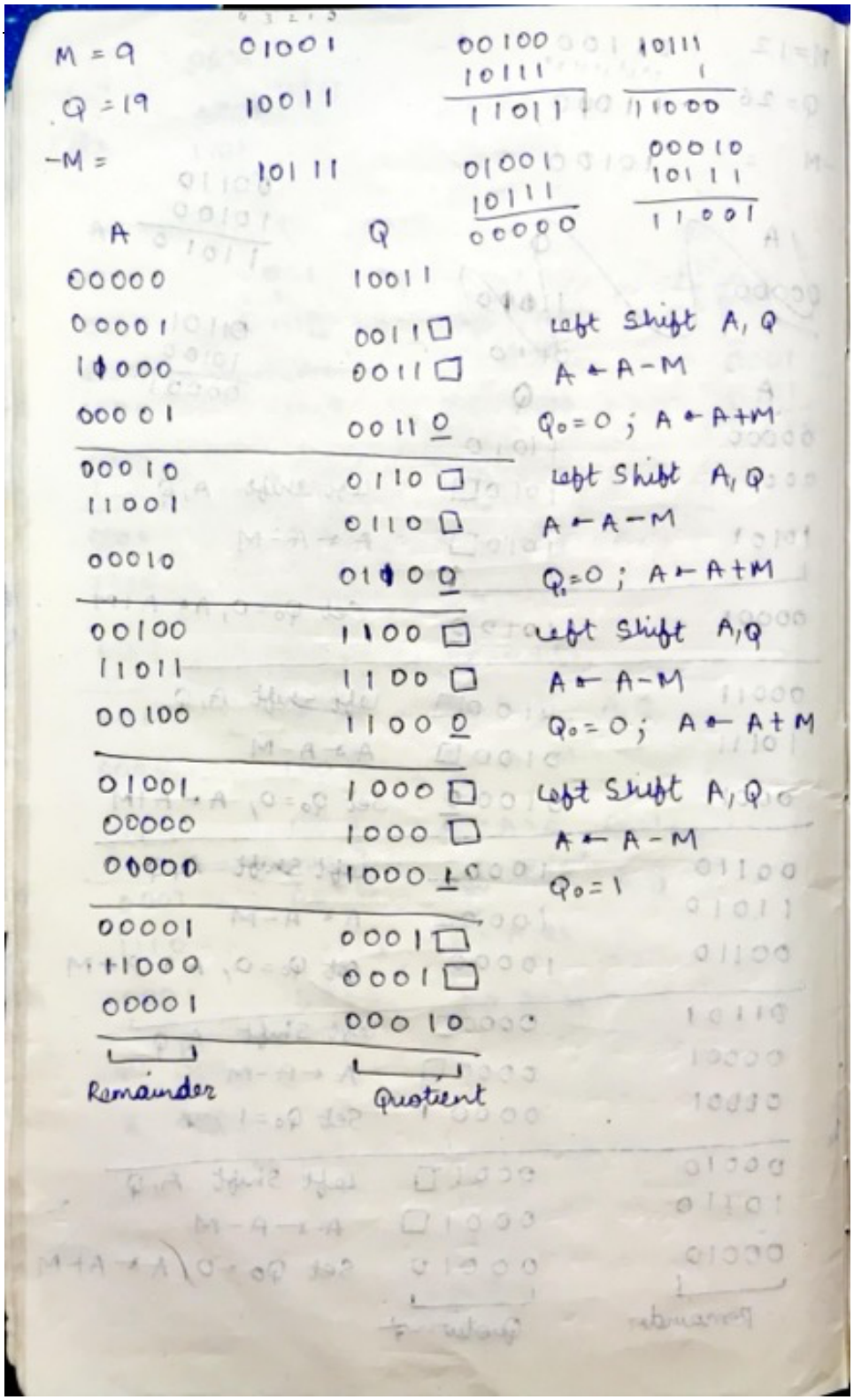
Flowchart for Restoring of Division:

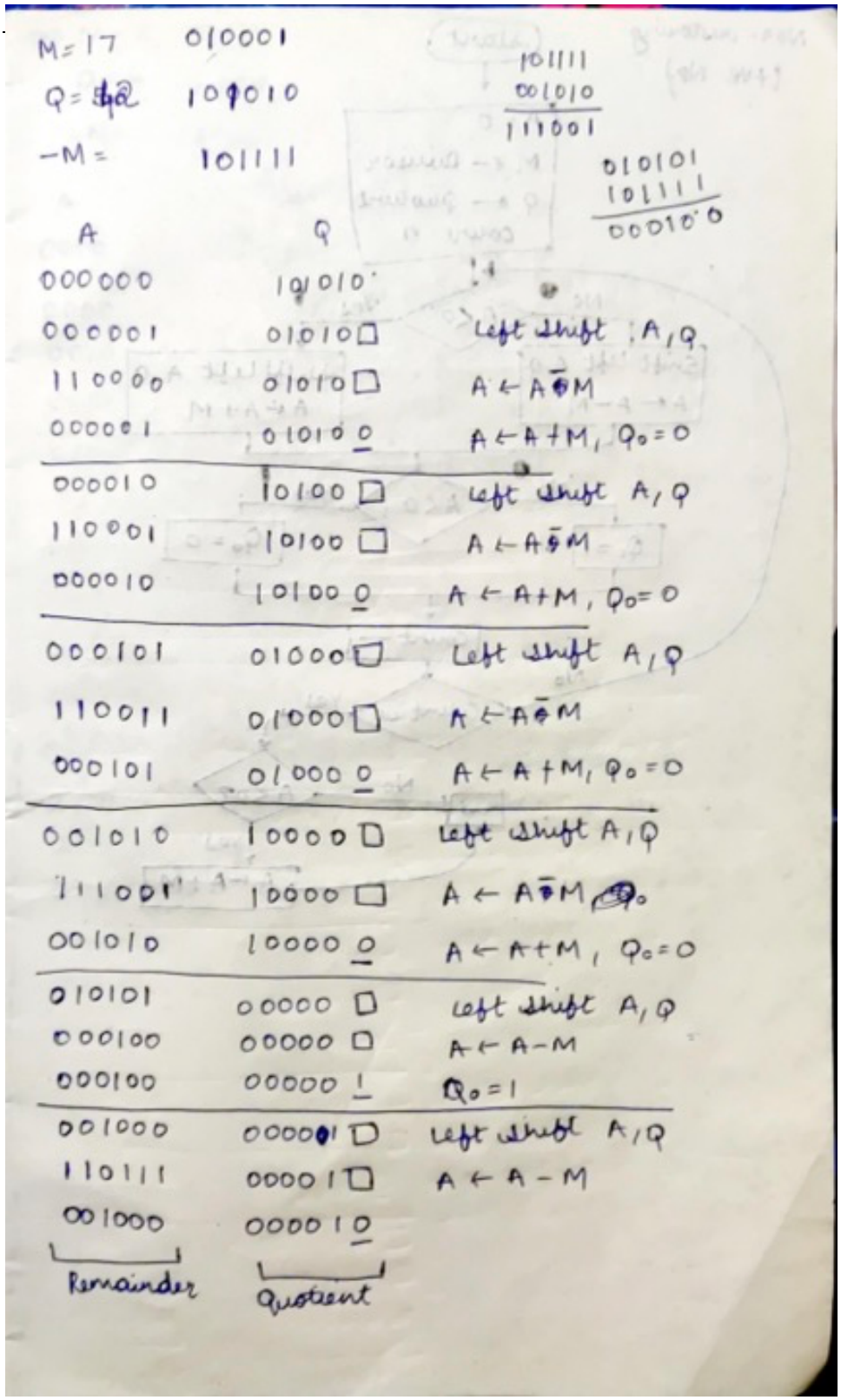


Design Steps:

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)





M = 17 010001
Q = 42 101010
−M = 101111

101111
001010
111001

010101
101111
000100

A	Q	
000000	101010	
000001	01010□	Left shift A, Q
110000	01010□	A ← A + M
000001	01010 0	A ← A + M, $Q_0 = 0$
000010	10100□	Left shift A, Q
110001	10100□	A ← A + M
000010	10100 0	A ← A + M, $Q_0 = 0$
000101	01000□	Left shift A, Q
110011	01000□	A ← A + M
000101	01000 0	A ← A + M, $Q_0 = 0$
001010	10000□	Left shift A, Q
111001	10000□	A ← A + M
001010	10000 0	A ← A + M, $Q_0 = 0$
010101	00000□	Left shift A, Q
000100	00000□	A ← A − M
000100	00000 1	$Q_0 = 1$
001000	00001□	Left shift A, Q
110111	00001□	A ← A − M
001000	00010	
Remainder	Quotient	

Code:

```
#include <bits/stdc++.h>
using namespace std;
int findbit(int m,int q){
    m=max(abs(m),abs(q));
    for(q=0;pow(2,q)<m;q++);
    return (max((q),4));
}
int* binary(int a,int num){
    int* ptr=(int*)malloc(num*sizeof(int));
    int acopy=abs(a),check=1;
    for (int i = 0;i<num; i++){
        ptr[i] = acopy % 2;
        acopy = acopy/2;
    }
    if (a < 0){
        for (int i = 0; i <num; i++){
            if (ptr[i] == 1 && check==1)
                check=0;
            else if(ptr[i] == 1 && check==0)
                ptr[i]=0;
            else if(ptr[i] == 0 && check==0)
                ptr[i]=1;
        }
    }
    return ptr;
}
void printbinary(int* ans,string s,int num){
    for(int i=(2*num)-1;i>num-1;i--)
        cout<<ans[i]<<" ";
    cout<<"\t";
    for(int i=num-1;i>=0;i--)
        cout<<ans[i]<<" ";
    cout<<"\t"<<s<<endl;
}
```

```
void binaryadd(int* ans,int* n,int num){
    int carry=0;
    for(int i=num;i<2*num;i++){
        if(ans[i]+n[i-num]+carry==1){
            ans[i]=1;
            carry=0;
        }
        else if(ans[i]+n[i-num]+carry==2){
            ans[i]=0;
            carry=1;
        }
        else if(ans[i]+n[i-num]+carry==3){
            ans[i]=1;
            carry=1;
        }
    }
}
int main()
{
    int m,q;
    cout<<"Enter Q and M: ";
    cin>>q>>m;
    int num=findbit(m,q);
    int ans[2*num]={0};
    int *arr=binary(q,num);
    for(int i=num-1;i>=0;i--)
        ans[i]=arr[i];
    cout<<endl<<"A\t\tQ\t\tOperation"<<endl<<endl;
    printbinary(ans,"Initial Value",num);
    for(int i=0;i<num;i++){
        for(int i=2*num;i>0;i--)
            ans[i]=ans[i-1]; // left Shifting
        ans[0]=8;
        printbinary(ans,"Arithemetic Shift Left",num);
        binaryadd(ans,binary(-m,num),num);
        printbinary(ans,"A <- A - M",num);
```

```
        if(ans[2*num-1]==1){
            binaryadd(ans,binary(m,num),num);
            printbinary(ans,"A <- A + M",num);
            ans[0]=0;
        }
        else
            ans[0]=1;
    }
    printbinary(ans,"Final Answer",num);
}
```

Output

```
pargat@Router Programs % cd "/Users/pargat/Documents/COL
rograms/"exp3
Enter Q and M: 19 9

A               Q               Operation

0 0 0 0 0       1 0 0 1 1       Initial Value
0 0 0 0 1       0 0 1 1 8       Arithemetic Shift Left
1 1 0 0 0       0 0 1 1 8       A <- A - M
0 0 0 0 1       0 0 1 1 8       A <- A + M
0 0 0 1 0       0 1 1 0 8       Arithemetic Shift Left
1 1 0 0 1       0 1 1 0 8       A <- A - M
0 0 0 1 0       0 1 1 0 8       A <- A + M
0 0 1 0 0       1 1 0 0 8       Arithemetic Shift Left
1 1 0 1 1       1 1 0 0 8       A <- A - M
0 0 1 0 0       1 1 0 0 8       A <- A + M
0 1 0 0 1       1 0 0 0 8       Arithemetic Shift Left
0 0 0 0 0       1 0 0 0 8       A <- A - M
0 0 0 0 1       0 0 0 1 8       Arithemetic Shift Left
1 1 0 0 0       0 0 0 1 8       A <- A - M
0 0 0 0 1       0 0 0 1 8       A <- A + M
0 0 0 0 1       0 0 0 1 0       Final Answer
pargat@Router Programs %
```

```
pargat@Router Programs % cd "/Users/pargat/Documents/COLLE
rograms/"exp3
Enter Q and M: 42 17

A               Q               Operation

0 0 0 0 0 0     1 0 1 0 1 0     Initial Value
0 0 0 0 0 1     0 1 0 1 0 8     Arithemetic Shift Left
1 1 0 0 0 0     0 1 0 1 0 8     A <- A - M
0 0 0 0 0 1     0 1 0 1 0 8     A <- A + M
0 0 0 0 1 0     1 0 1 0 0 8     Arithemetic Shift Left
1 1 0 0 0 1     1 0 1 0 0 8     A <- A - M
0 0 0 0 1 0     1 0 1 0 0 8     A <- A + M
0 0 0 1 0 1     0 1 0 0 0 8     Arithemetic Shift Left
1 1 0 1 0 0     0 1 0 0 0 8     A <- A - M
0 0 0 1 0 1     0 1 0 0 0 8     A <- A + M
0 0 1 0 1 0     1 0 0 0 0 8     Arithemetic Shift Left
1 1 1 0 0 1     1 0 0 0 0 8     A <- A - M
0 0 1 0 1 0     1 0 0 0 0 8     A <- A + M
0 1 0 1 0 1     0 0 0 0 0 8     Arithemetic Shift Left
0 0 0 1 0 0     0 0 0 0 0 8     A <- A - M
0 0 1 0 0 0     0 0 0 0 1 8     Arithemetic Shift Left
1 1 0 1 1 1     0 0 0 0 1 8     A <- A - M
0 0 1 0 0 0     0 0 0 0 1 8     A <- A + M
0 0 1 0 0 0     0 0 0 0 1 0     Final Answer
pargat@Router Programs %
```

Conclusion

The Restoring method of division has been studied and its implementation has been conducted successfully.

Post Lab Descriptive Questions

1. **What are the advantages of restoring division over non restoring division?**

In each step of your division calculation the result of the step is either 1 or 0, depending if the dividend is less than or larger than the divisor.
You generally do a test subtraction for each digit step; if the result is positive or zero, you note down a 1 as next digit of your quotient.
If the result is negative, you proceed with one of two strategies:
• restoring method: you add the divisor back, and put 0 as your next quotient digit
• non-restoring method: you don't do that - you keep negative remainder and a digit 1, and basically correct things by a supplementary addition afterwards.

Date: _____ **Signature of faculty in-charge**