Batch: B2	**Roll No.: 16010122151**
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)
CO1:Describe and define the structure of a computer with buses structure and detail working of the arithmetic logic unit and its sub modules

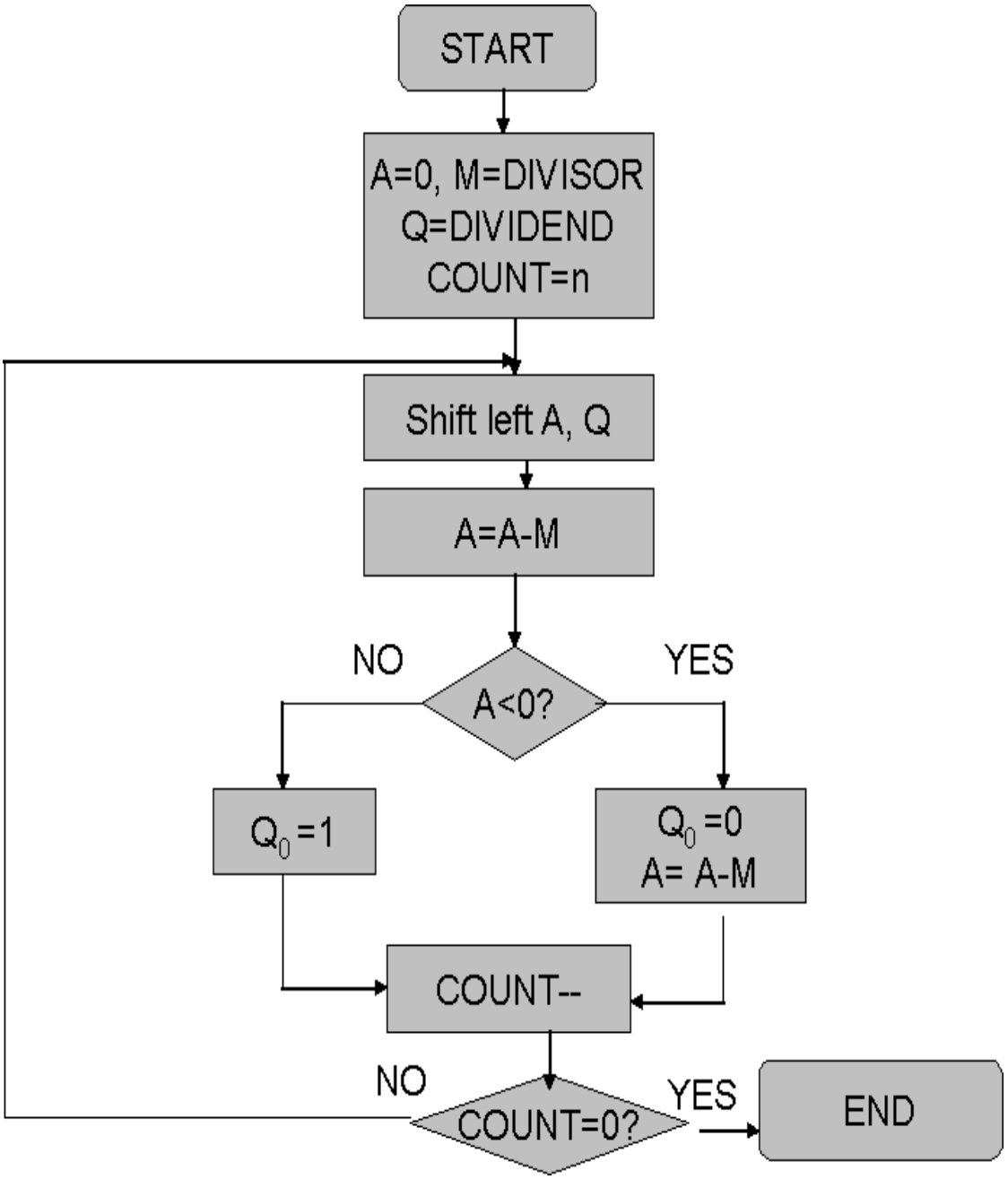
Books/ Journals/ Websites referred:

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

Code in C++:

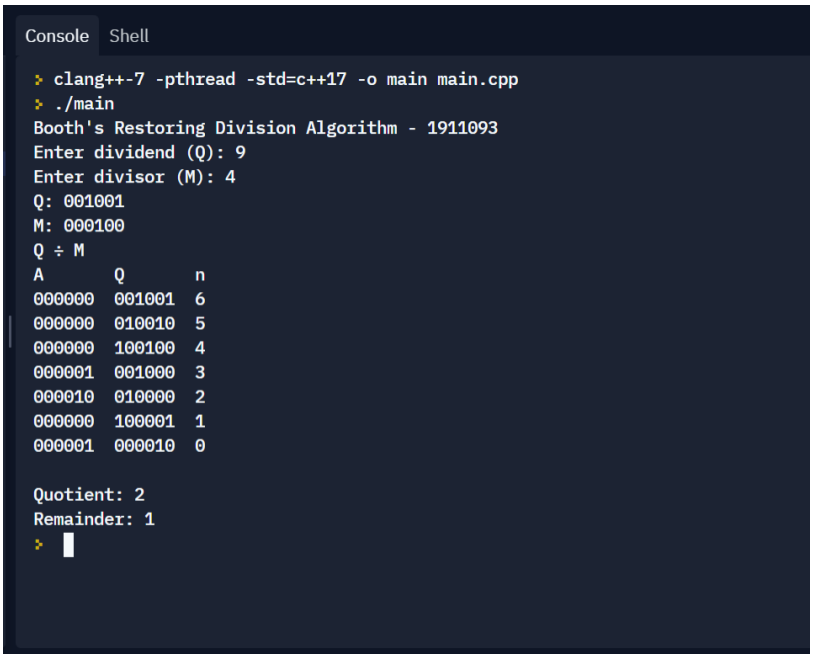
```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

void shiftLeft(vector<int> *A, vector<int> *Q){
  int temp = Q->front();
  rotate(A->begin(), A->begin() + 1, A->end());
  A->back() = temp;

  rotate(Q->begin(), Q->begin() + 1, Q->end());
}

vector<int> complement(vector<int> num){
  //1's complement
  for(int i = 0; i < num.size(); i++){
    if(num.at(i) == 0) num.at(i) = 1;
    else if(num.at(i) == 1) num.at(i) = 0;
  }
  //2's complement
  int carry = 0;
  for(int i = num.size() - 1; i >= 0; i--){
    if(num.at(i) == 0 && carry == 0){
      num.at(i) = 1;
      break;
    }
    else if(num.at(i) == 0 && carry == 1){
      num.at(i) = 1;
      break;
    }
    else if(num.at(i) == 1 && carry == 0){
      num.at(i) = 0;
      carry++;
    }
    else if(num.at(i) == 1 && carry == 1){
      num.at(i) = 0;
    }
  }
```

```
  return num;
}

void a_plus_m(vector<int> *a, vector<int> *m){
  vector<int> finalA;

  int sum = 0;
  for(int i = a->size() - 1; i >= 0; i--){
    if(a->at(i) == 1) sum++;
    if(m->at(i) == 1) sum++;

    if(sum == 1){
      finalA.push_back(1);
      sum = 0;
    }
    else if(sum == 3){
      finalA.push_back(1);
      sum = 1;
    }
    else if(sum == 2){
      finalA.push_back(0);
      sum = 1;
    }
    else finalA.push_back(0);
  }

  reverse(finalA.begin(), finalA.end());
  *a = finalA;
}

void a_minus_m(vector<int> *a, vector<int> *m){
  vector<int> newM = complement(*m);
  a_plus_m(a, &newM);
}

int getDecimal(vector<int> B){
  int sum = 0;
  for(int i = B.size() - 1; i >= 0; i--){
    sum += B.at(i) * pow(2, B.size() - 1 - i);
  }
  return sum;
}

int main() {
```

```
int n1, n2;

cout << "Booth's Restoring Division Algorithm - 1911093" << endl;
cout << "Enter dividend (Q): ";
cin >> n1;
cout << "Enter divisor (M): ";
cin >> n2;

//Convert to Binary
vector<int> Q, M;

while(n1 > 0){
  Q.push_back(n1 % 2);
  n1 /= 2;
}
while(Q.size() < 6) Q.push_back(0);
reverse(Q.begin(), Q.end());

while(n2 > 0){
  M.push_back(n2 % 2);
  n2 /= 2;
}
while(M.size() < 6) M.push_back(0);
reverse(M.begin(), M.end());

cout << "Q: ";
for(int n : Q) cout << n;
cout << endl;

cout << "M: ";
for(int n : M) cout << n;
cout << endl;
//End of conversion to Binary

cout << "Q ÷ M" << endl;

//Restoring Division Algorithm
vector<int> A = {0, 0, 0, 0, 0, 0};

cout << "A\t\tQ\t\tn" << endl;
for(int i = 0; i < 6; i++){
  //Printing
  for(int n : A) cout << n;
  cout << "\t";
```

```
    for(int n : Q) cout << n;
    cout << "\t";
    cout << 6 - i << endl;

    //Logic
    shiftLeft(&A, &Q);
    a_minus_m(&A, &M);

    //Check MSB of A
    if(A.front() == 1){ //Negative
      Q.back() = 0;
      a_plus_m(&A, &M);
    }
    else Q.back() = 1;
    //End of Logic
  }

  //Printing last row
  for(int n : A) cout << n;
  cout << "\t";
  for(int n : Q) cout << n;
  cout << "\t";
  cout << 0 << endl;

  int quotient = getDecimal(Q);
  int remain = getDecimal(A);

  cout << endl << "Quotient: " << quotient << endl;
  cout << "Remainder: " << remain << endl;

}
```

Output:


Console
Shell
clang++-7 -pthread -std=c++17 -o main main.cpp
./main
Booth's Restoring Division Algorithm - 1911093
Enter dividend (Q): 9
Enter divisor (M): 4
Q: 001001
M: 000100
Q ÷ M
A Q n
000000 001001 6
000000 010010 5
000000 100100 4
000001 001000 3
000010 010000 2
000000 100001 1
000001 000010 0
Quotient: 2
Remainder: 1

Conclusion: The concept of restoring method of division was used to divide two binary numbers and verified.

Post Lab Descriptive Questions

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

ANS: An extra bit must be maintained in the partial remainder to keep track of the sign in non-restoring division.

Date:17 - 8 - 23 **Signature of faculty in-charge**