UNSIGNED INTEGER DIVISION SIMULATION

MINIPROJECT REPORT

Submitted by

AMRITHA S (2020506011)

POOMALA M (2020506061)

SANDHIYA M (2020506078)

SHIFANI FRANCIS M (2020506086)

SOWBARNIGAA K S (2020506092)

MADRAS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UIVERSITY, CHENNAI- 600044

ABSTRACT

The application provides a visualization of the slow division algorithms. It displays the step by step approach in performing the division. The application prompts the user to enter the value for dividend and divisor in decimal form. The entered values are converted to binary form and the step by step execution of the algorithm is displayed. The result (quotient and remainder) is finally displayed in both decimal and binary form.

INTRODUCTION

PROBLEM STATEMENT:

A division algorithm is an algorithm which when given two integers dividend and divisor, computes their quotient and remainder. This is employed by digital circuit designs and software. Although it is a very basic operation, the hardware implementation of the division algorithm is a difficult task. Unlike what happens in software, where there is already a division operator, a hardware division module must be done with even more elementary calculations. Hence, understanding the algorithms can be difficult. An application that can make users understand these algorithms would be helpful.

OBJECTIVE OF THE PROJECT:

To develop an application that can perform restoring and non-restoring division with a visualization of each step in obtaining the result.

LANGUAGES USED

Figure 4.1

APPLICATION REQUIREMENTS

Figure 4.2

TWO APPROACHES TO THE PROBLEM

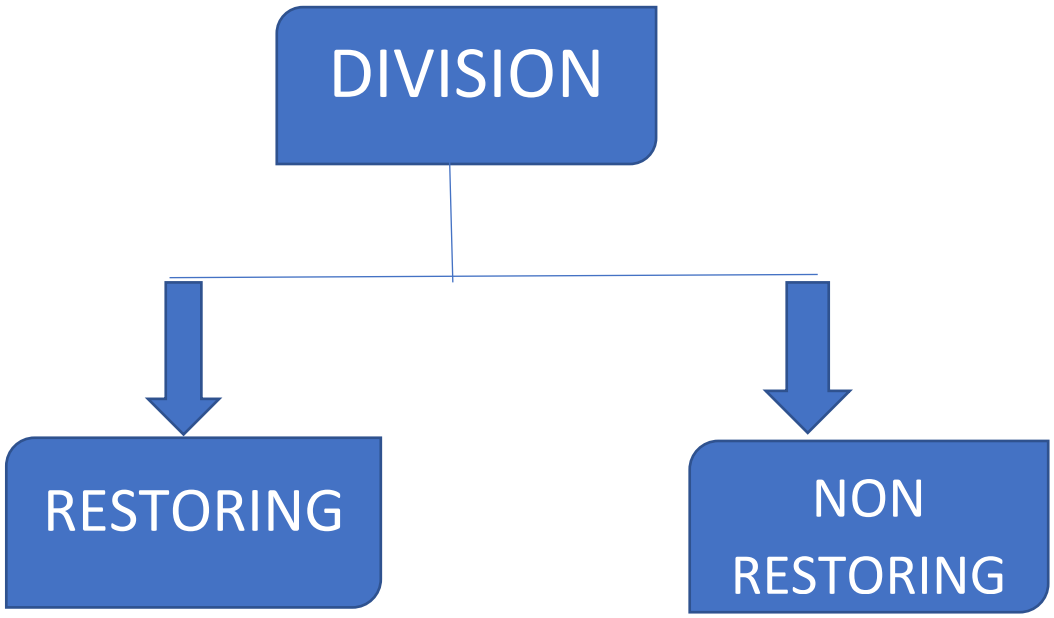


Figure 4.3

RESTORING DIVISION

Register Q contain quotient and register A contain remainder. Here, n-bit dividend is loaded in Q and divisor is loaded in M. Value of Register is initially kept 0 and this is the register whose value is restored during iteration due to which it is named Restoring.

STEPS INVOLVED

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Then the content of register A and Q is shifted left as if they are a single unit

Step-3: Then content of register M is subtracted from A and result is stored in A

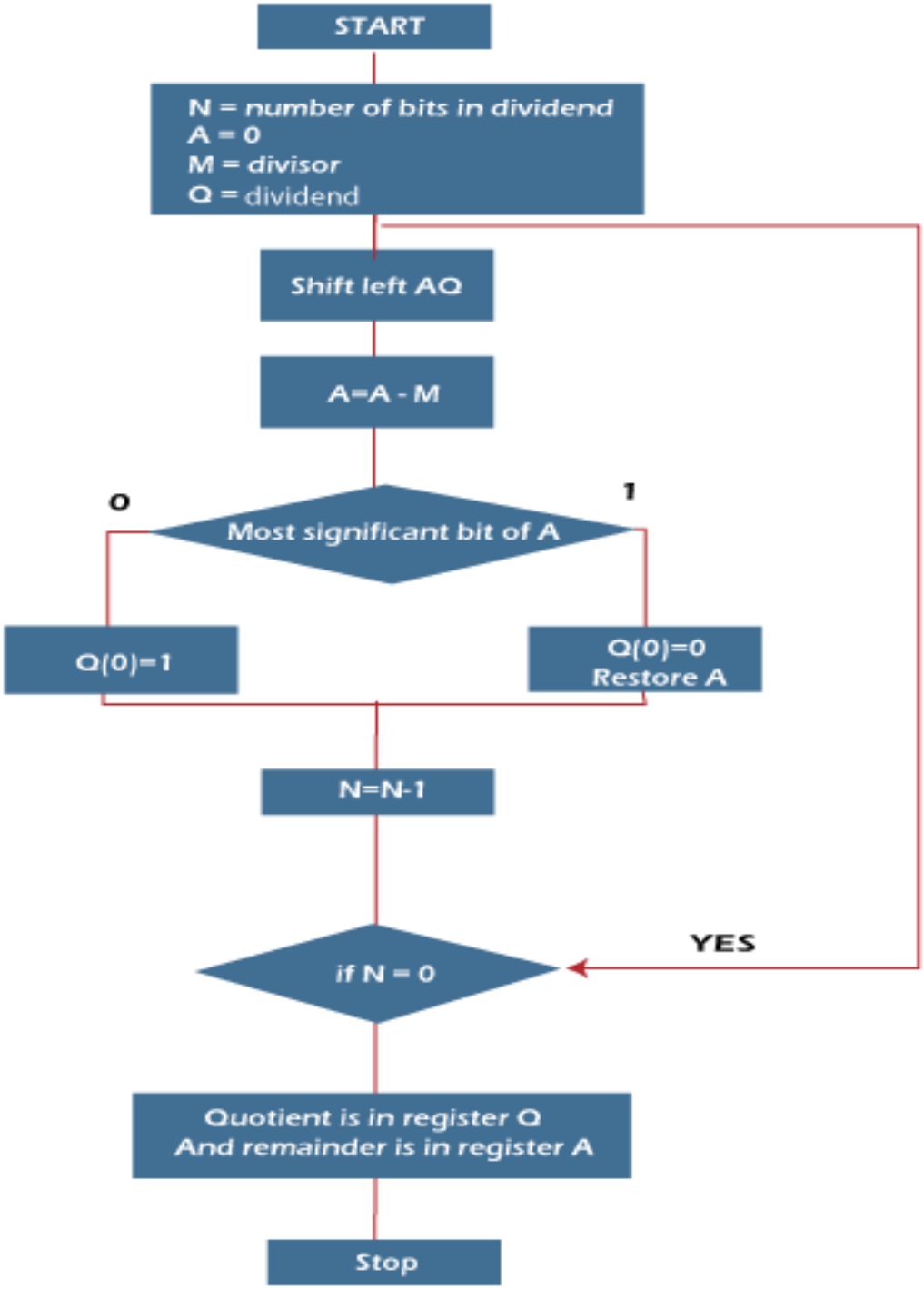
Step-4: Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit of Q is set to 0 and value of register A is restored i.e the value of A before the subtraction with M

Step-5: The value of counter n is decremented

Step-6: If the value of n becomes zero we get of the loop otherwise we repeat from step 2

Step-7: Finally, the register Q contain the quotient and A contain remainder

FLOW CHART


START
N = number of bits in dividend
A = 0
M = divisor
Q = dividend
Shift left AQ
A=A - M
0
Most significant bit of A
1
Q(0)=1
Q(0)=0
Restore A
N=N-1
YES
if N = 0
Quotient is in register Q
And remainder is in register A
Stop

IMPLEMENTATION

n	M	A	Q	Operation
4	00011	00000	1011	initialize
	00011	00001	011_	shift left AQ
	00011	11110	011_	A=A-M
	00011	00001	0110	Q[0]=0 And restore A
3	00011	00010	110_	shift left AQ
	00011	11111	110_	A=A-M
	00011	00010	1100	Q[0]=0
2	00011	00101	100_	shift left AQ
	00011	00010	100_	A=A-M
	00011	00010	1001	Q[0]=1
1	00011	00101	001_	shift left AQ
	00011	00010	001_	A=A-M
	00011	00010	0011	Q[0]=1

NON-RESTORING DIVISION

Register Q contains quotient, Register A contains remainder. Register Q contains n-bit dividend and M contains divisor. Value of Register is initially kept 0 and this is the register whose value is not restored during iteration Hence, named Non Restoring division.

STEPS INVOLVED

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Check the sign bit of register A

Step-3: If it is 1 shift left content of AQ and perform A = A+M, otherwise shift left AQ and perform A = A-M (means add 2's complement of M to A and store it to A)

Step-4: Again the sign bit of register A

Step-5: If sign bit is 1 Q[0] become 0 otherwise Q[0] become 1 (Q[0] means least significant bit of register Q)

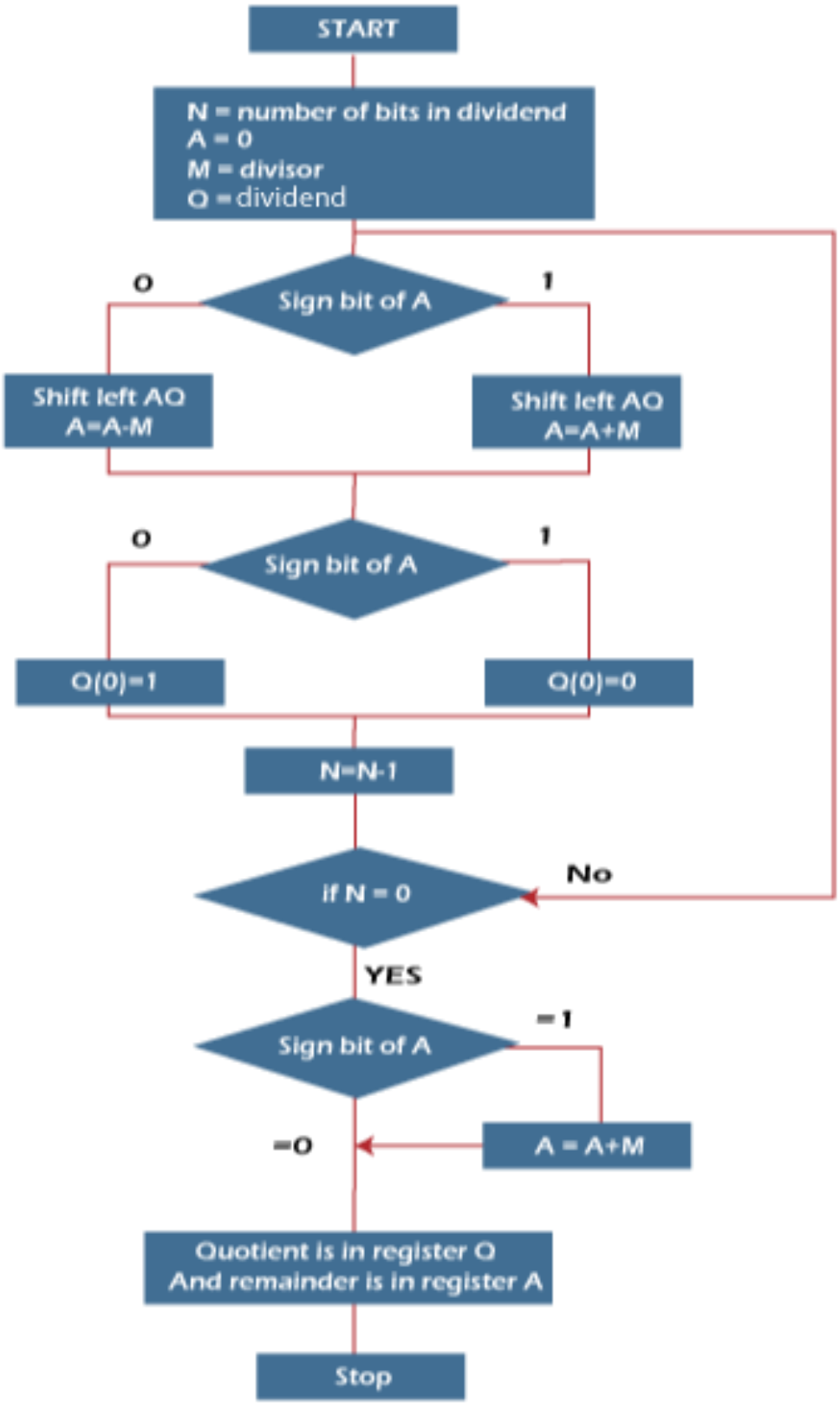
Step-6: Decrements value of N by 1

Step-7: If N is not equal to zero go to Step 2 otherwise go to next step

Step-8: If sign bit of A is 1 then perform A = A+M

Step-9: Register Q contain quotient and A contain remainder

FLOW CHART


START
N = number of bits in dividend
A = 0
M = divisor
Q = dividend
Sign bit of A
0
1
Shift left AQ
A=A-M
Shift left AQ
A=A+M
Sign bit of A
0
1
Q(0)=1
Q(0)=0
N=N-1
if N = 0
No
YES
Sign bit of A
=1
=0
A = A+M
Quotient is in register Q
And remainder is in register A
Stop

IMPLEMENTATION

N	M	A	Q	Action
4	00011	00000	1011	Start
		00001	011_	Left shift AQ
		11110	011_	A=A-M
3		11110	0110	Q[0]=0
		11100	110_	Left shift AQ
		11111	110_	A=A+M
2		11111	1100	Q[0]=0
		11111	100_	Left Shift AQ
		00010	100_	A=A+M
1		00010	1001	Q[0]=1
		00101	001_	Left Shift AQ
		00010	001_	A=A-M
0		00010	0011	Q[0]=1

ALGORITHM

A class was created to perform non restoring division. Class member variables are dividend, divisor, remainder, quotient. Member functions are created to binary addition to find 2's compliment of a binary number, to convert a number from decimal to binary form, to perform left shift, to display each step in the implementation, to convert a number from binary form to decimal form and to perform division.

A class was created to perform restoring division. Class member variables are dividend, divisor, remainder, quotient. Member functions are created to binary addition, to find 2's compliment of a binary number, to convert a number from decimal to binary form, to perform left shift, to display each step in the implementation, to convert a number from binary form to decimal form and to perform division.

Objects are created for both restoring and non restoring division classes. A menu is displayed to the user and the users are prompted to choose which type of division is to be performed. According to the choice given by the user, the corresponding functions are called.

PROGRAM

```
#include<bits/stdc++.h>
using namespace std;
class  NonResDiv{
public:
    string a;
string q;
string m;
string mc;
int rem;
int quotient;
string bin_add(string s1,string s2){
    string ans=(s1.length()>=s2.length())?s1:s2;
    int c=0;
    int i=s1.length()-1;
```

```
    int j=s2.length()-1;
    int k=ans.size()-1;
    //cout<<"Ans: "<<ans<<" "<<endl;
    while(i>=0 and j>=0){
      // cout<<"a: "<<s1[i]<<" "<<"b: "<<s2[j]<<"c:
"<<c<<endl;
        if(c==1){
            if(s1[i]=='0' and s2[j]=='0')
                  {
                    c=0;
                    ans[k]='1';
                  }
            else if (s1[i]=='1' and s2[j]=='1')
            {
                c=1;
                ans[k]='1';
            }
            else
            {
                c=1;
                ans[k]='0';
            }
                    }
        else{
            if(s1[i]=='0' and s2[j]=='0')
```

```
            {
              c=0;
              ans[k]='0';
            }
        else if (s1[i]=='1' and s2[j]=='1')
        {
            c=1;
            ans[k]='0';
        }
        else
        {
            c=0;
            ans[k]='1';
        }
    }
    i--;
    j--;
    k--;
        //cout<<"Ans: "<<ans<<" "<<endl;
    }
    while(k!=0 and c!=0){
        if(s1[i]=='1'){
            ans[k]='0';
            c=1;
        }
```

```
            else{
                ans[k]='1';
                c=0;
            }
            k--;
            i--;
        }
        return ans;

}
string complement_2(string m)
{
    for(int i=0;i<m.size();i++){
        m[i]=(m[i]-'0'==0)?'1':'0';
    }

    string ans=bin_add(m,"1");
    return ans;
}
string to_binary(int d){
    string bin="";
    while(d>0){
        bin.insert(0,to_string(d%2));
        d/=2;
    }
```

```
    return bin;
}
void shift_left(){
    for(int i=0;i<a.size()-1;i++){
        a[i]=a[i+1];
    }
    a[a.size()-1]=q[0];
    for(int i=0;i<q.size()-1;i++){
       q[i]=q[i+1];
    }
    q[q.size()-1]='_';
//  cout<<" A:   "<<a<<" "<<q<<endl;
}
void divide(){
    cout<<" A:   "<<a<<" "<<q<<endl<<endl;
    for(int i=0;i<q.size();i++){
        int s=a[0];
        shift_left();
        if(s=='0'){
            cout<<"A-M(since A[0]=0)"<<endl;
            cout<<" A:   "<<a<<" "<<q<<endl;
            cout<<"-M:   "<<mc<<endl;
            a=bin_add(a,mc);
        }
        else{
```

```
            cout<<"A+M(since A[1]=1)"<<endl;
            cout<<" A:   "<<a<<" "<<q<<endl;
            cout<<"+M:   "<<m<<endl;
            a=bin_add(a,m);
        }
        if(a[0]=='1')
        {
            q[q.size()-1]='0';
        }
        else{
            q[q.size()-1]='1';
        }
        cout<<"----------------------"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        cout<<"----------------------"<<endl;
        cout<<endl;

    }
    if(a[0]=='1'){
        a=bin_add(a,m);
        cout<<"----------------------"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        cout<<"----------------------"<<endl;
    }
```

```
}
   void bin_to_decimal(){
        rem=0;
        quotient=0;
        int n=a.length();
        int m=q.length();
        int dummy=1;
        for(int i=n-1;i>=0;i--){
           if(a[i]=='1'){
             rem+=dummy;
                 }
             dummy*=2;
         }
         dummy=1;
         for(int i=m-1;i>=0;i--){
           if(q[i]=='1'){
             quotient+=dummy;
                 }
             dummy*=2;
         }
         cout<<"\nRemainder: "<<rem<<"(A:
"<<a<<")"<<endl;
         cout<<"Quotient: "<<quotient<<"(Q:
"<<q<<")"<<endl;
```

```
   }
    void perform_division(){

printf("\n\t\t\t``````````````````````````````````````
``````````````````");
    printf("\n\t\t\t\t`
************************************** `");
    printf("\n\t\t\t\t` *   NON-RESTORING
DIVISION   * `");
    printf("\n\t\t\t\t`
************************************** `");
    printf("\n\t\t\t\t``````````````````````````````````
``````````````````");
    cout<<"\nSteps involved in Non -Restoring
division........"<<endl;
    cout<<"\nStep-1: First the registers are
initialized with corresponding values (Q =
Dividend, M = Divisor, A = 0, n = number of bits in
dividend)"<<endl;
    cout<<"\nStep-2: Then the most significant bit
of the A is checked if it is 0 then Add M else add
-M"<<endl;
    cout<<"\nStep-3: Then the content of register A
and Q is shifted left as if they are a single
unit"<<endl;
    cout<<"\nStep-4: Then the most significant bit
of the A is checked if it is 0 the least
significant bit of Q is set to 1 otherwise if it is
1 the least significant bit of Q is set to 0 and
```

```
value of register A is restored i.e the value of A
before the subtraction with M"<<endl;

    cout<<"\nStep-5: The value of counter n is
decremented"<<endl;

    cout<<"\nStep-6: If the value of n becomes zero
we get of the loop otherwise we repeat from step
2"<<endl;

    cout<<"\nStep-7: Finally, the register Q
contain the quotient and A contain
remainder"<<endl;

    cout<<endl;

    int dividend,divisor;

    cout<<"Enter the dividend: ";

    cin>>dividend;

    cout<<"\nEnter the divisor: ";

    cin>>divisor;

        if(divisor==0){

        cout<<"No you can't divide by 0.."<<endl;

        return;

    }

    string divi_d=to_binary(dividend);//covert
given to binary format

    string divi_r=to_binary(divisor);

    //cout<<divi_d<<"\n"<<divi_r<<endl;

    divi_r.insert(0,to_string(0));//add sign bit
for because a and m should be fo same length

    //divi_d.insert(0,to_string(0));
```

```
    /*cout<<divi_d<<"\n"<<divi_r<<endl;
    string a=bin_add(divi_d,divi_r)  ;
    cout<<a<<endl;*/
    a="";
    for(int i=0;i<divi_r.size();i++){
        a+='0';
    }
    m=divi_r;
    q=divi_d;
    mc=complement_2(m);
    cout<<"\nBasic Calculations..."<<endl;
    cout<<"**********************************"<<en
dl;
    cout<<"*         -M : "<<mc<<"
"<<endl;
    cout<<"*          Q : "<<q<<"
"<<endl;
    cout<<"*          M : "<<m<<"
"<<endl;
    cout<<"* Iterations : "<<q.size()<<"
"<<endl;
cout<<"**********************************"<<endl;
    cout<<"\nSTEP BY STEP
EXECUTION........."<<endl;
    divide();
    bin_to_decimal();
    rem=0;
```

```
    quotient=0;
    }
};
class ResDiv{
public:
    string a;
string q;
string m;
string mc;
int rem;
int quotient;
string bin_add(string s1,string s2){
    string ans=(s1.length()>=s2.length())?s1:s2;
    int c=0;
    int i=s1.length()-1;
    int j=s2.length()-1;
    int k=ans.size()-1;
    //cout<<"Ans: "<<ans<<" "<<endl;
    while(i>=0 and j>=0){
      // cout<<"a: "<<s1[i]<<" "<<"b: "<<s2[j]<<"c:
"<<c<<endl;
        if(c==1){
            if(s1[i]=='0' and s2[j]=='0')
                  {
                    c=0;
```

```
                ans[k]='1';
              }
        else if (s1[i]=='1' and s2[j]=='1')
        {
            c=1;
            ans[k]='1';
        }
        else
        {
            c=1;
            ans[k]='0';
        }

    }
    else{
        if(s1[i]=='0' and s2[j]=='0')
              {
                c=0;
                ans[k]='0';
              }
        else if (s1[i]=='1' and s2[j]=='1')
        {
            c=1;
            ans[k]='0';
        }
```

```
            else
            {
                c=0;
                ans[k]='1';
            }
        }
        i--;
        j--;
        k--;
            //cout<<"Ans: "<<ans<<" "<<endl;
        }
        while(k!=0 and c!=0){
            if(s1[i]=='1'){
                ans[k]='0';
                c=1;
            }
            else{
                ans[k]='1';
                c=0;
            }
            k--;
            i--;
        }
        return ans;
}
```

```
string complement_2(string m)
{
    for(int i=0;i<m.size();i++){
        m[i]=(m[i]-'0'==0)?'1':'0';
    }

    string ans=bin_add(m,"1");
    return ans;
}
string to_binary(int d){
    string bin="";
    while(d>0){
        bin.insert(0,to_string(d%2));
        d/=2;
    }
    return bin;
}
void shift_left(){
    for(int i=0;i<a.size()-1;i++){
        a[i]=a[i+1];
    }
    a[a.size()-1]=q[0];
    for(int i=0;i<q.size()-1;i++){
       q[i]=q[i+1];
    }
```

```
    q[q.size()-1]='_';
//  cout<<" A:   "<<a<<" "<<q<<endl;
}
void divide(){
            cout<<" A:   "<<a<<" "<<q<<endl;
    for(int i=0;i<q.size();i++){
        cout<<"******Iteration:
"<<i+1<<"********"<<endl;
        int s=a[0];
        shift_left();
        cout<<"A-M"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        cout<<"-M:   "<<mc<<endl;
        a=bin_add(a,mc);
        cout<<"-----------------------"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        if(a[0]=='1')
        {
            q[q.size()-1]='0';
            cout<<"A+M(since A[0]=1)"<<endl;
            cout<<" A:   "<<a<<" "<<q<<endl;
            cout<<"+M:   "<<m<<" "<<endl;
            a=bin_add(a,m);
        }
        else{
```

```
            q[q.size()-1]='1';
        }
        cout<<"----------------------"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        cout<<"----------------------"<<endl;
        cout<<endl;
        cout<<endl;
    }
    if(a[0]=='1'){
        cout<<"A+M(since A[0]=1)"<<endl;
        a=bin_add(a,m);
        cout<<"----------------------"<<endl;
        cout<<" A:   "<<a<<" "<<q<<endl;
        cout<<"----------------------"<<endl;
    }
}
 void bin_to_decimal(){
       rem=0;
       quotient=0;
       int n=a.length();
       int m=q.length();
       int dummy=1;
       for(int i=n-1;i>0;i--){
          if(a[i]=='1'){
            rem+=dummy;
```

```
                }
            dummy*=2;
        }
        dummy=1;
        for(int i=m-1;i>=0;i--){
          if(q[i]=='1'){
            quotient+=dummy;
                }
            dummy*=2;
        }
        cout<<"\nRemainder: "<<rem<<"(A:
"<<a<<")"<<endl;
        cout<<"Quotient: "<<quotient<<"(Q:
"<<q<<")"<<endl;
   }
     void perform_division(){

printf("\n\t\t\t``````````````````````````````````
````````````````");
    printf("\n\t\t\t`
**************************************  `");
    printf("\n\t\t\t`  *            RESTORING
DIVISION            *  `");
    printf("\n\t\t\t`
**************************************  `");
    printf("\n\t\t\t``````````````````````````````````
````````````````");
```

```
    cout<<"\nSteps involved in Restoring
division........"<<endl;

    cout<<"\nStep-1: First the registers are
initialized with corresponding values (Q =
Dividend, M = Divisor, A = 0, n = number of bits in
dividend)"<<endl;

    cout<<"\nStep-2: Then the content of register A
and Q is shifted left as if they are a single
unit"<<endl;

    cout<<"\nStep-3: Then content of register M is
subtracted from A and result is stored in
A."<<endl;

    cout<<"\nStep-4: Then the most significant bit
of the A is checked if it is 0 the least
significant bit of Q is set to 1 otherwise if it is
1 the least significant bit of Q is set to 0 and
value of register A is restored i.e the value of A
before the subtraction with M"<<endl;

    cout<<"\nStep-5: The value of counter n is
decremented"<<endl;

    cout<<"\nStep-6: If the value of n becomes zero
we get of the loop otherwise we repeat from step
2"<<endl;

    cout<<"\nStep-7: Finally, the register Q
contain the quotient and A contain
remainder"<<endl;

    cout<<endl;

    int dividend,divisor;

    cout<<"Enter the dividend: ";

    cin>>dividend;
```

```
    cout<<"\nEnter the divisor: ";
    cin>>divisor;
    if(divisor==0){
        cout<<"No you can't divide by 0.."<<endl;
        return;
    }
    string divi_d=to_binary(dividend);//covert
given to binary format
    string divi_r=to_binary(divisor);
    //cout<<divi_d<<"\n"<<divi_r<<endl;
    divi_r.insert(0,to_string(0));//add sign bit
for a because a and m should be of same length
    //divi_d.insert(0,to_string(0));
    /*cout<<divi_d<<"\n"<<divi_r<<endl;
    string a=bin_add(divi_d,divi_r)  ;
    cout<<a<<endl;*/
    a="";
    for(int i=0;i<divi_r.size();i++){
        a+='0';
    }
    m=divi_r;
    q=divi_d;

    mc=complement_2(m);
    cout<<"\nBasic Calculations..."<<endl;
```

```
    cout<<"*********************************"<<en
dl;
    cout<<"*         -M : "<<mc<<"
"<<endl;
    cout<<"*          Q : "<<q<<"
"<<endl;
    cout<<"*          M : "<<m<<"
"<<endl;
    cout<<"* Iterations : "<<q.size()<<"
"<<endl;

cout<<"*********************************"<<endl;
    cout<<"\nSTEP BY STEP
EXECUTION........."<<endl;
    divide();
    bin_to_decimal();
    rem=0;
    quotient=0;
     }
};
int main(){
    printf("\n\t\t\t`````````````````````````````````
`````````````````````");
    printf("\n\t\t\t`
************************************** `");
    printf("\n\t\t\t`   *   UNSIGNED INTEGER
DIVSION SIMULATION   * `");
```
```

```
    printf("\n\t\t\t`
**************************************  `");
    printf("\n\t\t\t`````````````````````````````
`````````````````````");

printf("\n\t\t\t**********************************
*****************");
    printf("\n\t\t\t*   GENERAL
INSTRUCTION   *");
    printf("\n\t\t\t*
*");
    printf("\n\t\t\t*   1.Enter the only
integers.   *");
    printf("\n\t\t\t*   2.-----represents result
*");
    printf("\n\t\t\t*   3.Don't give other
numbers apart menu.   *");
    printf("\n\t\t\t*
*");
    printf("\n\t\t\t**********************************
**********************");
    printf("\n->->A division algorithm provides a
quotient and a remainder when we divide two number.
They are generally of two type slow algorithm and
fast algorithm. Slow division algorithm are
restoring, non-restoring, non-performing restoring,
SRT algorithm and under fast comes Newton–Raphson
and Goldschmidt.");
```
```

```
    int c;


        ResDiv d;
        NonResDiv r;
    do{

    printf("\n\n\t\t\t~~~~~~~~~~~~~~~~~~~~HOME~~~~~~
~~~~~~~~~~~~~~~");

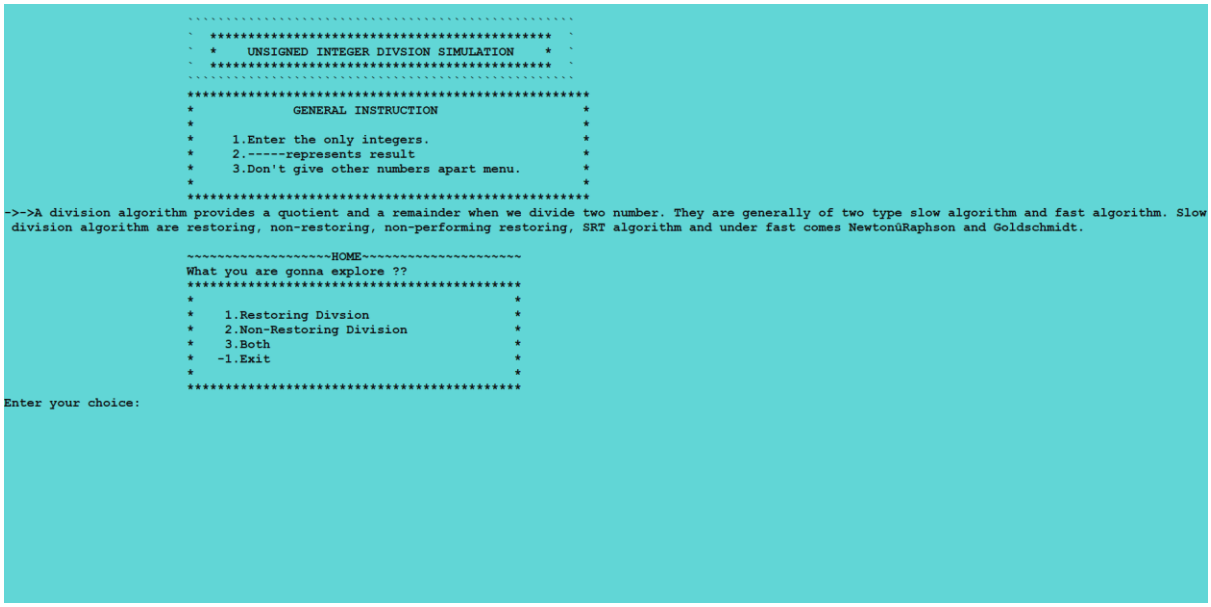

    printf("\n\t\t\tWhat you are gonna explore
??");
    printf("\n\t\t\t**********************************
************");
    printf("\n\t\t\t*
*");
    printf("\n\t\t\t*    1.Restoring Divsion
*");
    printf("\n\t\t\t*    2.Non-Restoring Division
*");
    printf("\n\t\t\t*    3.Both
*");
    printf("\n\t\t\t*   -1.Exit
*");
    printf("\n\t\t\t*
*");
    printf("\n\t\t\t**********************************
************");
        cout<<"\nEnter your choice: ";
```

```
    cin>>c;
    switch(c){
        case 1:
            d.perform_division();
            break;
        case 2:
            r.perform_division();
            break;
        case 3:
            d.perform_division();
            r.perform_division();
            break;
        case -1:
                printf("\a");

    printf("\n\t\t\t**********************************
**********");
                printf("\n\t\t\t*   Thank you for
visiting\1!! See Again!!  *");

    printf("\n\t\t\t**********************************
**********");
                exit(1);
                break;
    }}while(c!=-1);
}
```

OUTPUT SCREENSHOT



```
Enter the dividend: 5

Enter the divisor: 3

Basic Calculations...
***********************************
*         -M : 101
*          Q : 101
*          M : 011
* Iterations : 3
***********************************

STEP BY STEP EXECUTION.........
 A:   000 101
******Iteration: 1********
A-M
 A:   001 01_
-M:   101
----------------------
 A:   110 01_
A+M(since A[0]=1)
 A:   110 010
+M:   011
----------------------
 A:   001 010
----------------------


******Iteration: 2********
A-M
 A:   010 10_
-M:   101
----------------------
 A:   111 10_
A+M(since A[0]=1)
 A:   111 100
+M:   011
----------------------
 A:   010 100
----------------------
```

```
******Iteration: 2********
A-M
 A:    010 10_
-M:    101
-----------------------
 A:    111 10_
A+M(since A[0]=1)
 A:    111 100
+M:    011
-----------------------
 A:    010 100
-----------------------


******Iteration: 3********
A-M
 A:    101 00_
-M:    101
-----------------------
 A:    010 00_
-----------------------
 A:    010 001
-----------------------



Remainder: 2(A: 010)
Quotient: 1(Q: 001)


                        ~~~~~~~~~~~~~~~~~~~~HOME~~~~~~~~~~~~~~~~~~~~~~
                        What you are gonna explore ??
                        ********************************************
                        *                                          *
                        *     1.Restoring Divsion                  *
                        *     2.Non-Restoring Division             *
                        *     3.Both                               *
                        *    -1.Exit                               *
                        *                                          *
                        ********************************************
Enter your choice:
```

```
                    ``````````````````````````````````````````````````
                    `  ****************************************  `
                    `  *          NON-RESTORING DIVISION        *  `
                    `  ****************************************  `
                    ``````````````````````````````````````````````````
Steps involved in Non -Restoring division........

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Then the most significant bit of the A is checked if it is 0 then Add M else add -M

Step-3: Then the content of register A and Q is shifted left as if they are a single unit

Step-4: Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit
 of Q is set to 0 and value of register A is restored i.e the value of A before the subtraction with M

Step-5: The value of counter n is decremented

Step-6: If the value of n becomes zero we get of the loop otherwise we repeat from step 2

Step-7: Finally, the register Q contain the quotient and A contain remainder

Enter the dividend:
```

```
Enter the dividend: 5

Enter the divisor: 3

Basic Calculations...
***********************************
*         -M : 101
*          Q : 101
*          M : 011
* Iterations : 3
***********************************

STEP BY STEP EXECUTION.........
 A:   000 101

A-M(since A[0]=0)
 A:   001 01_
-M:   101
-----------------------
 A:   110 010
-----------------------

A+M(since A[1]=1)
 A:   100 10_
+M:   011
-----------------------
 A:   111 100
-----------------------

A+M(since A[1]=1)
 A:   111 00_
+M:   011
-----------------------
 A:   010 001
-----------------------


Remainder: 2(A: 010)
Quotient: 1(Q: 001)
```

```
                    ~~~~~~~~~~~~~~~~~~~HOME~~~~~~~~~~~~~~~~~~~~
                    What you are gonna explore ??
                    *******************************************
                    *                                         *
                    *    1.Restoring Divsion                  *
                    *    2.Non-Restoring Division             *
                    *    3.Both                               *
                    *   -1.Exit                               *
                    *                                         *
                    *******************************************
Enter your choice: -1

                    *******************************************
                    *   Thank you for visiting☺!! See Again!!  *
                    *******************************************
--------------------------------
Process exited after 149.9 seconds with return value 1
Press any key to continue . . .
```

## RESULT

Thus, an application that can perform restoring and non-restoring division along with providing a visualization of the steps involved was successfully developed.