**NAME** :- Manish Shashikant Jadhav

**UID** :- 2023301005.

**BRANCH** :- Comps -B. **BRANCH:** B.

**EXPERIMENT 7: Implement Restroing Division Algorithm.**

**SUBJECT** :- CAO (COMPUTER ARCHITECTURE AND ORGANIZATION)

**CODE** :- #include <iostream>

#include <string>

#include <algorithm>

using namespace std;

// Function to add two binary numbers

string add(string A, string M) {

    int carry = 0;

    string Sum;

    // Iterating through the number

    // A. Here, it is assumed that

    // the length of both the numbers

    // is same

    for (int i = A.length() - 1; i >= 0; i--) {

        //Adding the values at both

        // the indices along with the

        //carry

        int temp = A[i] - '0' + M[i] - '0' + carry;

        // If the binary number exceeds 1

        if (temp > 1) {

            Sum.push\_back('0' + (temp % 2));

            carry = 1;

        }

        else {

            Sum.push\_back('0' + temp);

            carry = 0;

        }

    }

    // Return the sum from Most Significant to Low Significant

    reverse(Sum.begin(), Sum.end());

    return Sum;

}

// Function of find the complement of the binary number

string complement(string m) {

    string M;

    // Iterating through the number

    for (int i = 0; i < m.length(); i++) {

        // Computing the Complement

        M.push\_back('0' + ((m[i] - '0' + 1) % 2));

    }

    // Adding 1 to the computed value

    M = add(M, "0001");

    return M;

}

// Function to find the quotient and remainder

// Using Restoring Division

void restoringDivision(string Q, string M, string A) {

    int count = M.length();

    //Printing the initial values

    // of the accumulator, dividend

    // and divisor

    cout << "Initial Values: A:" << A << " Q:" << Q << " M:" << M << endl;

    // The number of steps is equal to the

    // length of the binary number

    while (count > 0) {

        // Printing the values at every step

        cout << "\nstep:" << (M.length() - count + 1) << endl;

        A = A.substr(1) + Q[0];

        // Taking complement and adding it to A

        // Indirectly we are subtracting(A-M)

        string comp\_M = complement(M);

        A = add(A, comp\_M);

        // Left shift,assigning LSB of Q to MSB of A.

        cout << "Left Shift and Subtract: ";

        cout << " A:" << A << endl;

        cout << "A:" << A << " Q:" << Q.substr(1) << "\_";

        if (A[0] == '1') {

            // Unsuccessful and Quotient bit will be zero

            Q = Q.substr(1) + '0';

            cout << " -Unsuccessful" << endl;

            // Restoration is required for A

            A = add(A, M);

            cout << "A:" << A << " Q:" << Q << " -Restoration" << endl;

        }

        else {

            // Quotient bit will be 1

            Q = Q.substr(1) + '1';

            cout << " Successful" << endl;

            // No restoration

            cout << "A:" << A << " Q:" << Q << " -No Restoration" << endl;

        }

        count--;

    }

    // Final quotient and remainder of the

    // given dividend and divisor

    cout << "\nQuotient(Q):" << Q << " Remainder(A):" << A << endl;

}

int main() {

    string dividend = "0101";

    string divisor = "0111";

    string accumulator

            = string(dividend.length(), '0');

    restoringDivision(dividend, divisor, accumulator);

    return 0;

}

**OUTPUT** :-

