

| **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. Q0=0 & store present A
9. Else Q0=0 & restore previous A
10. Decrement count.
11. If count=0 go to 11
12. Else go to 3
13. STOP

Code:  
#include <iostream>

#include <bitset>

#include <string>

using namespace std;

string int2binary(int a) {

bitset<6> bs(a);

return bs.to\_string();

}

int binary2int(string a) {

int result = 0;

for (char bit : a) {

result = (result << 1) | (bit - '0');

}

return result;

}

string addbinary(const string& bin1, const string& bin2) {

int carry = 0;

string result;

for (int i = 5; i >= 0; --i) {

int bit1 = bin1[i] - '0';

int bit2 = bin2[i] - '0';

int sum = bit1 + bit2 + carry;

result = char((sum % 2) + '0') + result;

carry = sum / 2;

}

return result;

}

string twosComplement(const string& binary) {

string onesComplement;

for (char bit : binary) {

onesComplement += (bit == '0') ? '1' : '0';

}

return addbinary(onesComplement, "000001");

}

void leftShift(string& a, string& q, string& bm\_twosComplement) {

a = a.substr(1,6) + q.front();

q = q.substr(1,6) + '\_';

a = addbinary(a , bm\_twosComplement);

}

int main() {

int m, q, n = 6;

string a = "000000";

cout << "Om Thanage\t16010123217\n";

cout << "Welcome to Restoring Division Algorithm (upto 6 bits)\n";

cout << "Enter Divisor M: ";

cin >> m;

cout << "Enter Dividend Q: ";

cin >> q;

string bm = int2binary(m);

string bq = int2binary(q);

string bm\_twosComplement = twosComplement(bm);

cout << "Binary of M: " << bm << endl;

cout << "Binary of Q: " << bq << endl;

cout << "2's Complement of M: " << bm\_twosComplement << endl;

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

cout << "A: " << a << ", Q: " << bq << "\n\n";

while (n--) {

cout<<"\n Cycle:"<<6-n<<endl;

leftShift(a, bq, bm\_twosComplement);

cout<<" A : "<<a<<"\t Q : "<<bq<<"\tLeft Shift\t=>\tA-M\n";

if(a.front() == '1'){

bq = bq.substr(0,5)+"0";

a = addbinary(a, bm);

cout<<" A : "<<a<<"\t Q : "<<bq<<"\tSet Q₀ = 0 \t=>\tA+M\n";

}

else{ bq = bq.substr(0,5)+"1"; cout<<" A : "<<a<<"\t Q : "<<bq<<"\tSet Q₀ = 1\n";}

}

int Quotient = binary2int(bq);

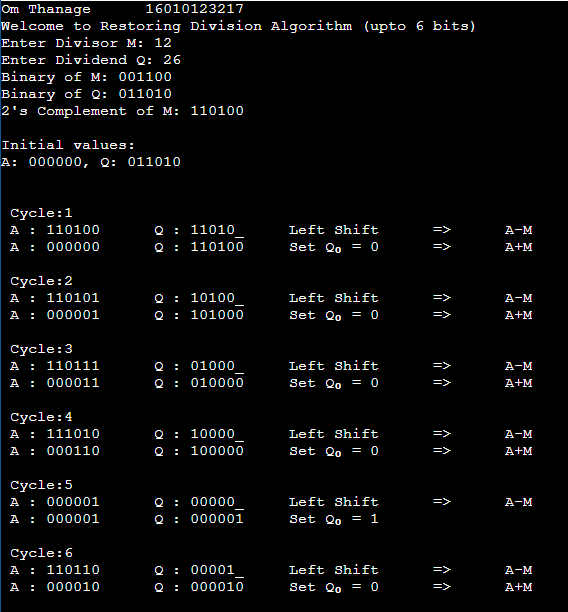
int Remainder = binary2int(a);

cout << "\nQuotient: " << Quotient << "\nRemainder: " << Remainder << endl;

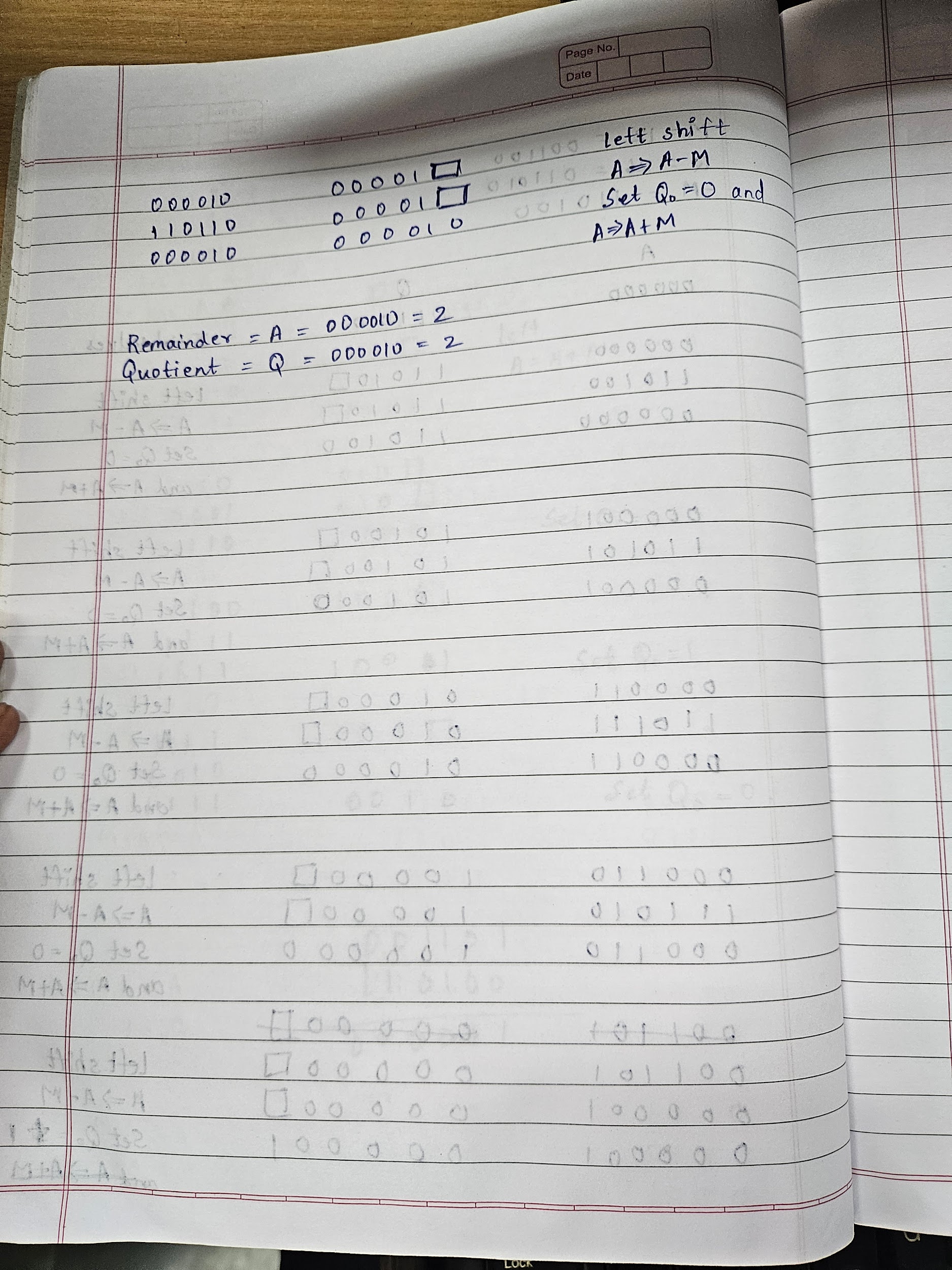
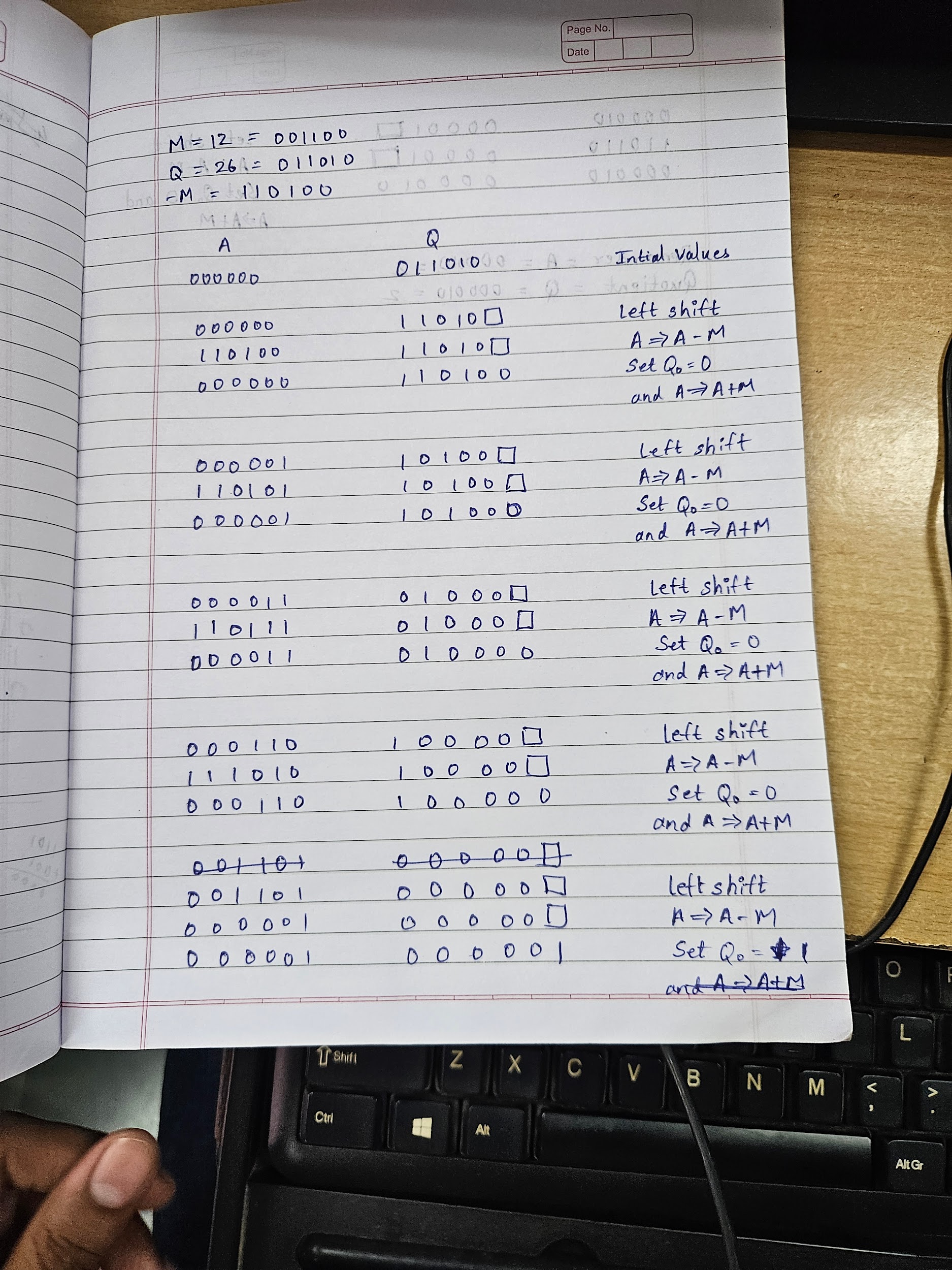
return 0;

}

Output:


**Example:- (Handwritten solved problems needs to be uploaded)**



**Conclusion**

**By this doing this experiment we learnt an efficient algorithm for division in which the dividend is restored whenever necessary**

**Post Lab Descriptive Questions**

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

**Ans. 1) Restoring division is easy to understand and implement**

**2)Hardware compatible**

**3)Accurate results**

**Date: 16/08/24**