**Batch: A3 Roll No.:16010121051**

**Experiment / assignment / tutorial No.\_\_\_\_\_\_\_**

**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.

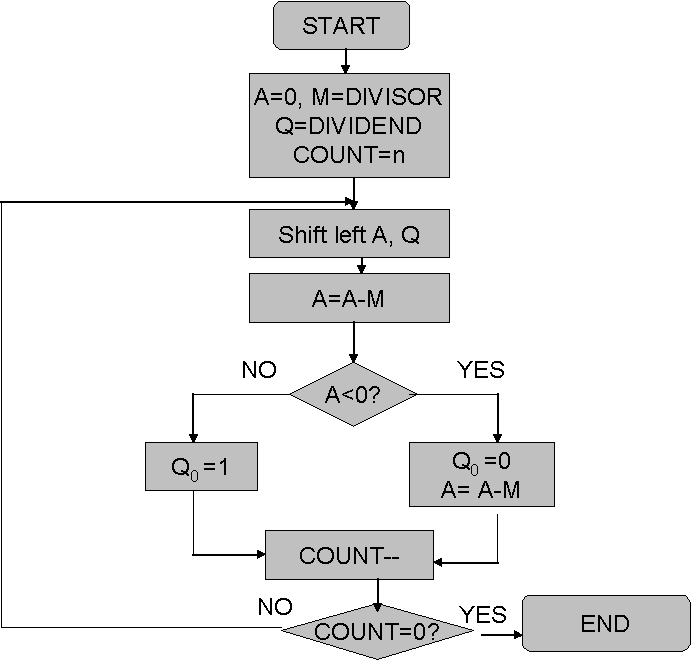
**3**. Dr. M. Usha, T. S. Srikanth, “Computer System Architecture and Organization”, First Edition, Wiley-India.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

The Restoring algorithm works with any combination of positive and negative numbers.

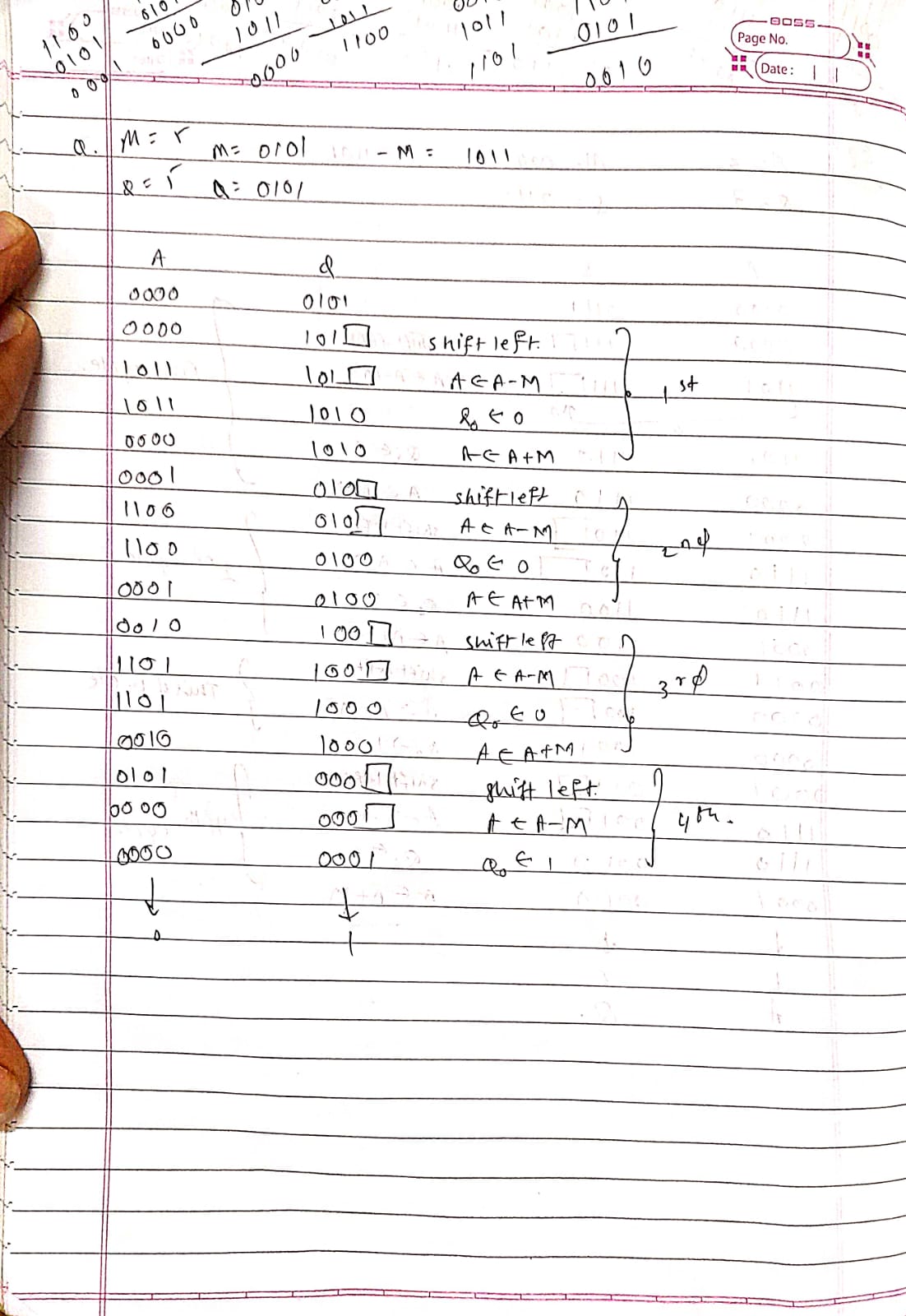
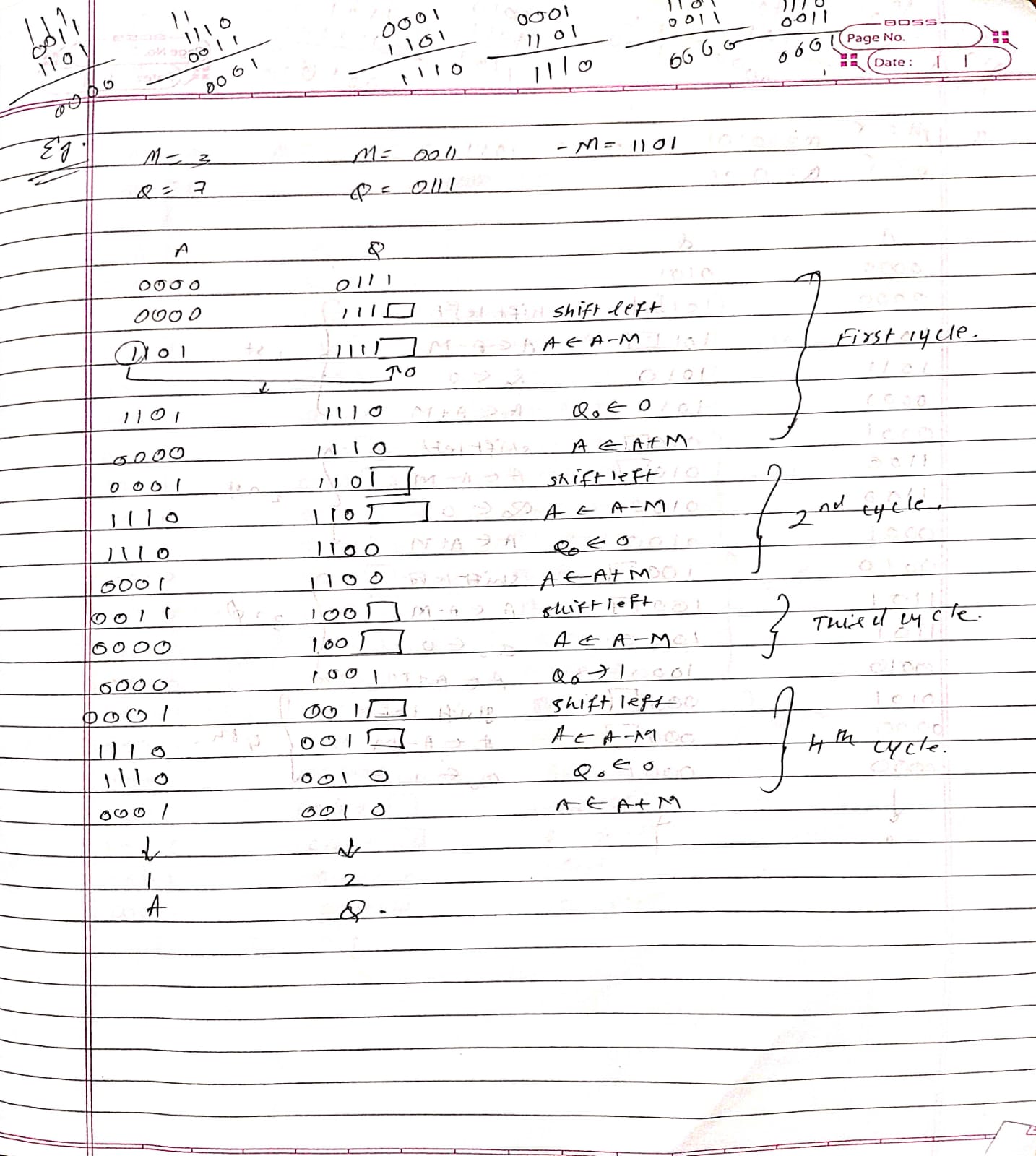
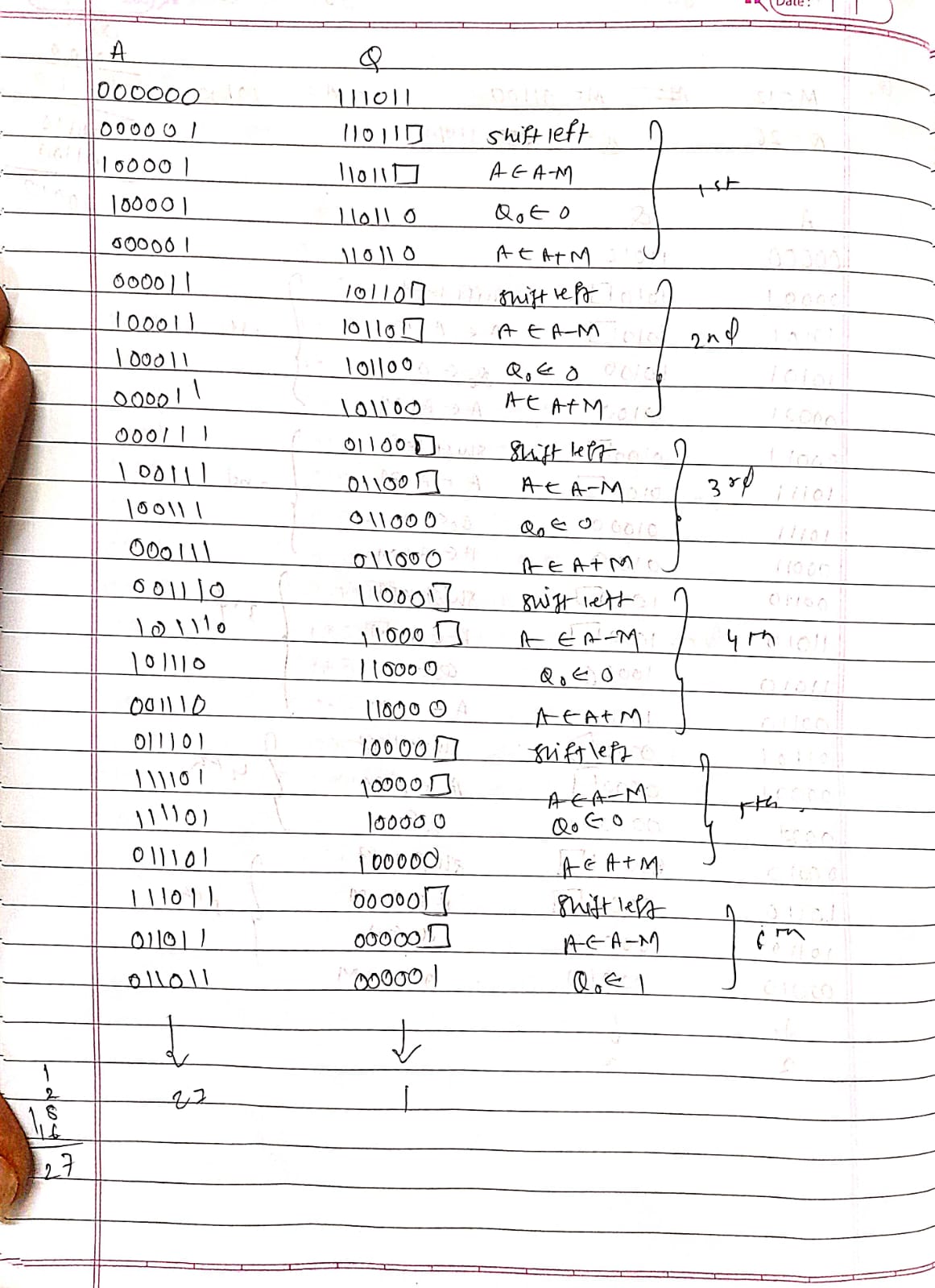
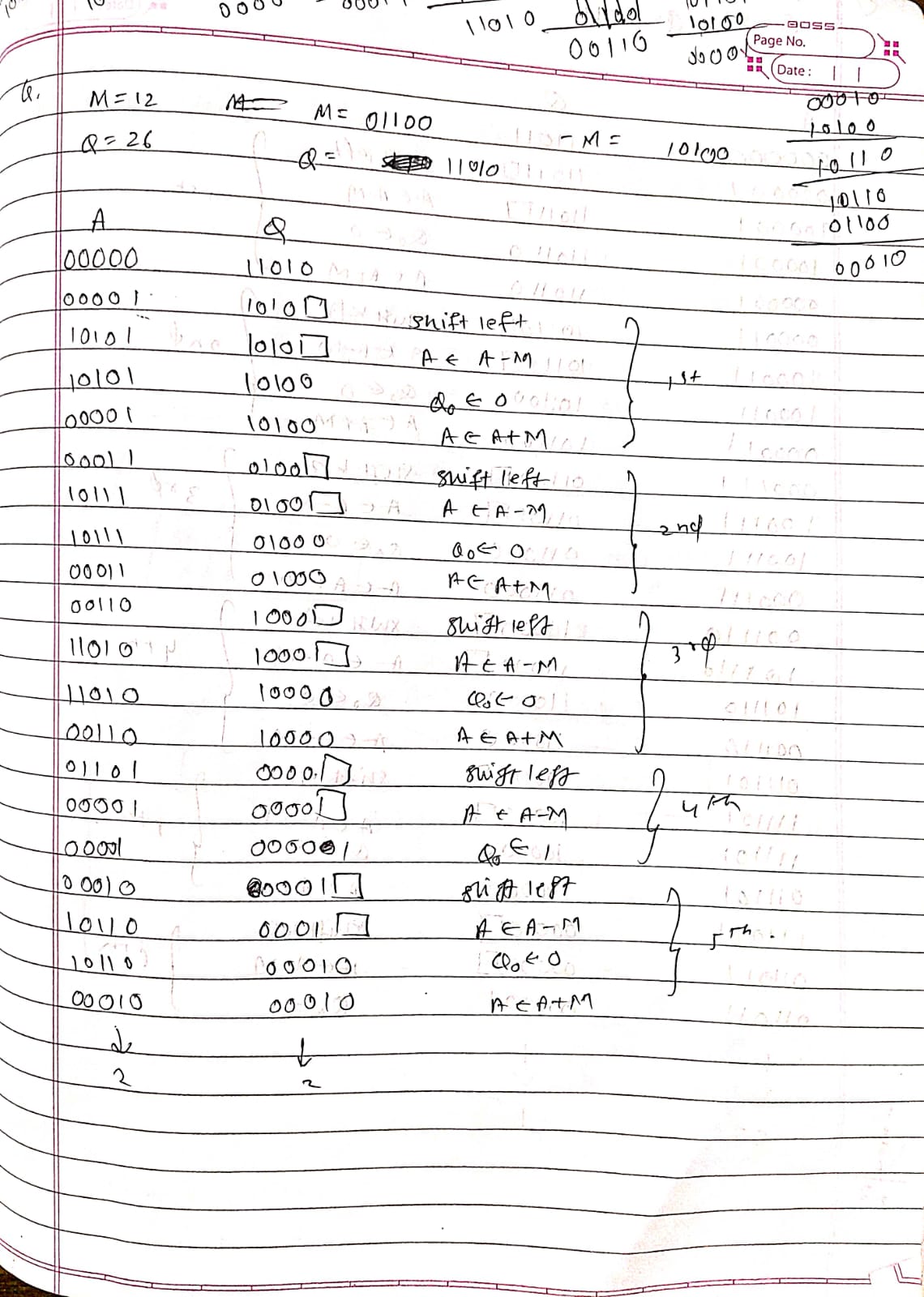
**Flowchart for Restoring of Division:**



**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. 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

**Example:-**



Implementation:

/\*Program for Restoring Division.\*/

#include <stdio.h>

#include <conio.h>

#include <math.h>

int a=0,b=0,c=0,com[5]={1,0,0,0,0},s=0;

int anum[5]={0},anumcp[5] ={0},bnum[5]={0};

int acomp[5]={0},bcomp[5]={0},rem[5]={0},quo[5]={0},res[5]={0};

void binary(){

a = fabs(a);

b = fabs(b);

int r, r2, i, temp;

for(i = 0; i < 5; i++){

r = a % 2;

a = a / 2;

r2 = b % 2;

b = b / 2;

anum[i] = r;

anumcp[i] = r;

bnum[i] = r2;

if(r2 == 0){

bcomp[i] = 1;

}

if(r == 0){

acomp[i] =1;

}

}

//part for two's complementing

c = 0;

for( i = 0; i < 5; i++){

res[i] = com[i]+ bcomp[i] + c;

if(res[i]>=2){

c = 1;

}

else

c = 0;

res[i] = res[i]%2;

}

for(i = 4; i>= 0; i--){

bcomp[i] = res[i];

}

}

void add(int num[]){

int i;

c = 0;

for( i = 0; i < 5; i++){

res[i] = rem[i]+ num[i] + c;

if(res[i]>=2){

c = 1;

}

else

c = 0;

res[i] = res[i]%2;

}

for(i = 4; i>= 0; i--){

rem[i] = res[i];

printf("%d",rem[i]);

}

printf(":");

for(i = 4; i>= 0; i--){

printf("%d",anumcp[i]);

}

}

void shl(){//for shift left

int i;

for(i = 4; i > 0 ; i--){//shift the remainder

rem[i] = rem[i-1];

}

rem[0] = anumcp[4];

for(i = 4; i > 0 ; i--){//shift the remtient

anumcp[i] = anumcp[i-1];

}

anumcp[0] = 0;

printf("\nSHIFT LEFT: ");//display together

for(i = 4; i>= 0; i--){

printf("%d",rem[i]);

}

printf(":");

for(i = 4; i>= 0; i--){

printf("%d",anumcp[i]);

}

}

int main(){

;

int i;

printf("\t\tRESTORING DIVISION ALGORITHM");

printf("\nEnter two numbers to multiply: ");

printf("\nBoth must be less than 16");

//simulating for two numbers each below 16

do{

printf("\nEnter Dividened: ");

scanf("%d",&a);

printf("Enter Divisor: ");

scanf("%d",&b);

}while(a>=16 || b>=16);

printf("\nExpected Quotient = %d", a/b);

printf("\nExpected Remainder = %d", a%b);

if(a\*b <0){

s = 1;

}

binary();

printf("\n\nUnsigned Binary Equivalents are: ");

printf("\nA = ");

for(i = 4; i>= 0; i--){

printf("%d",anum[i]);

}

printf("\nB = ");

for(i = 4; i>= 0; i--){

printf("%d",bnum[i]);

}

printf("\nB'+ 1 = ");

for(i = 4; i>= 0; i--){

printf("%d",bcomp[i]);

}

printf("\n\n-->");

//division part

shl();

for(i=0;i<5;i++){

printf("\n-->"); //start with subtraction

printf("\nSUB B: ");

add(bcomp);

if(rem[4]==1){//simply add for restoring

printf("\n-->RESTORE");

printf("\nADD B: ");

anumcp[0] = 0;

add(bnum);

}

else{

anumcp[0] = 1;

}

if(i<4)

shl();

}

printf("\n----------------------------");

printf("\nSign of the result = %d",s);

printf("\nRemainder is = ");

for(i = 4; i>= 0; i--){

printf("%d",rem[i]);

}

printf("\nQuotient is = ");

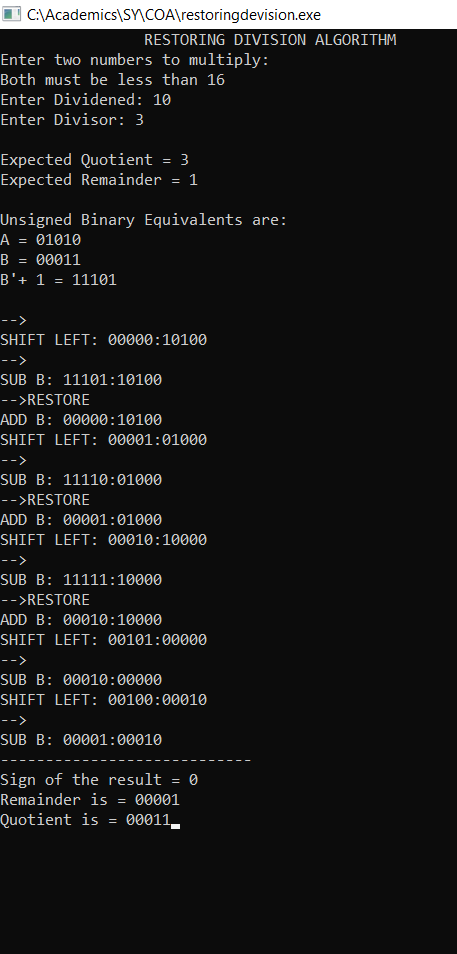
for(i = 4; i>= 0; i--){

printf("%d",anumcp[i]);

}

getch();

}

Output:

**Conclusion:**

**Booths restoring division was implemented successfully.**

**Post Lab Descriptive Questions**

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

The advantage of using non - restoring arithmetic over the standard restoring division is that a test subtraction is not required; the sign bit determines whether an addition or subtraction is used. The disadvantage, though, is that an extra bit must be maintained in the partial remainder to keep track of the sign.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**