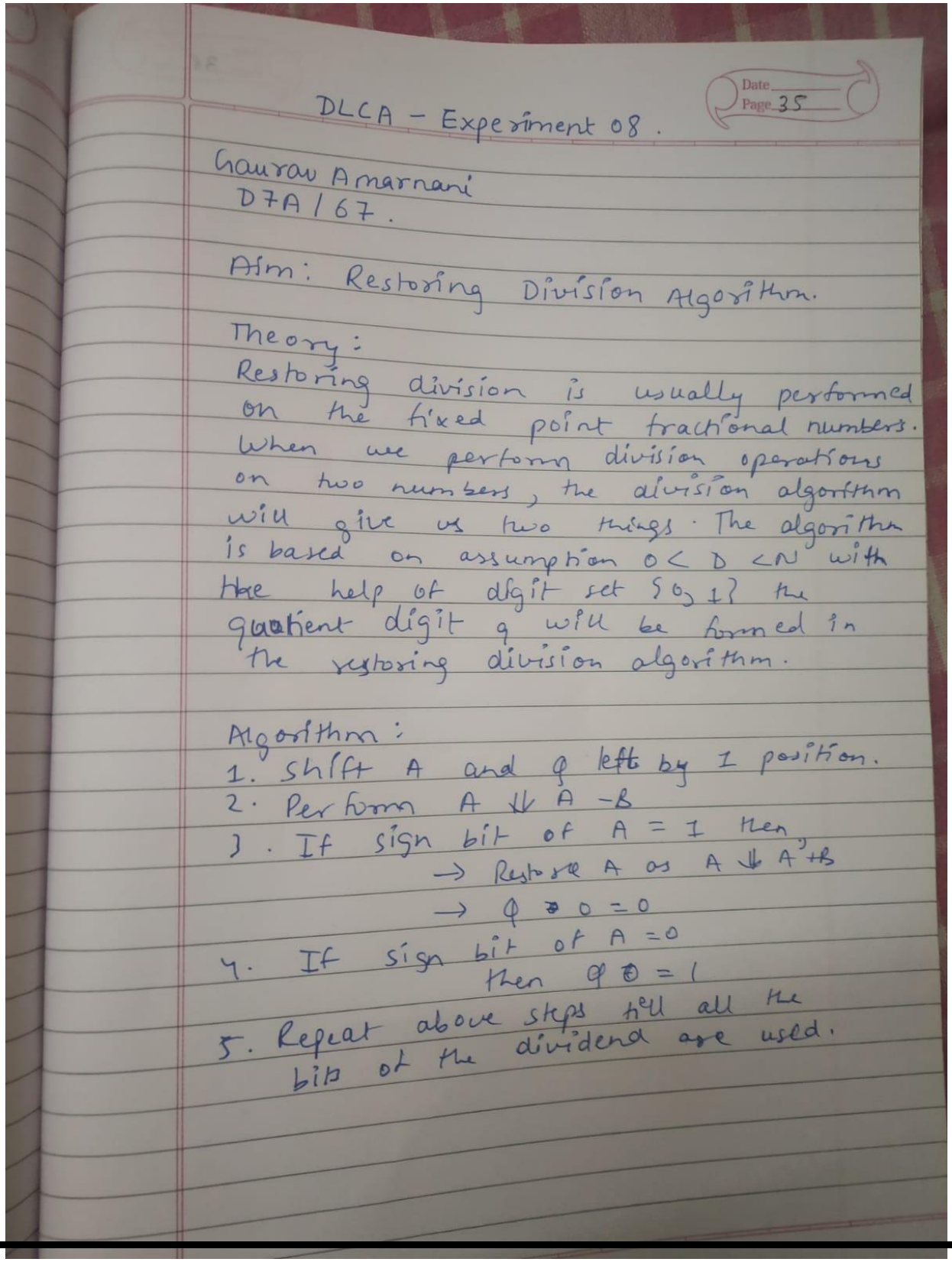
COMPUTER ENGINEERING

DLCA ODD SEM 2021-22/EXPERIMENT 8

NAME:- GAURAV AMARNANI (D7A, 67)



DLCA – Experiment 08.

Date
Page 35

Gaurav Amarnani
D7A / 67.

Aim: Restoring Division Algorithm.

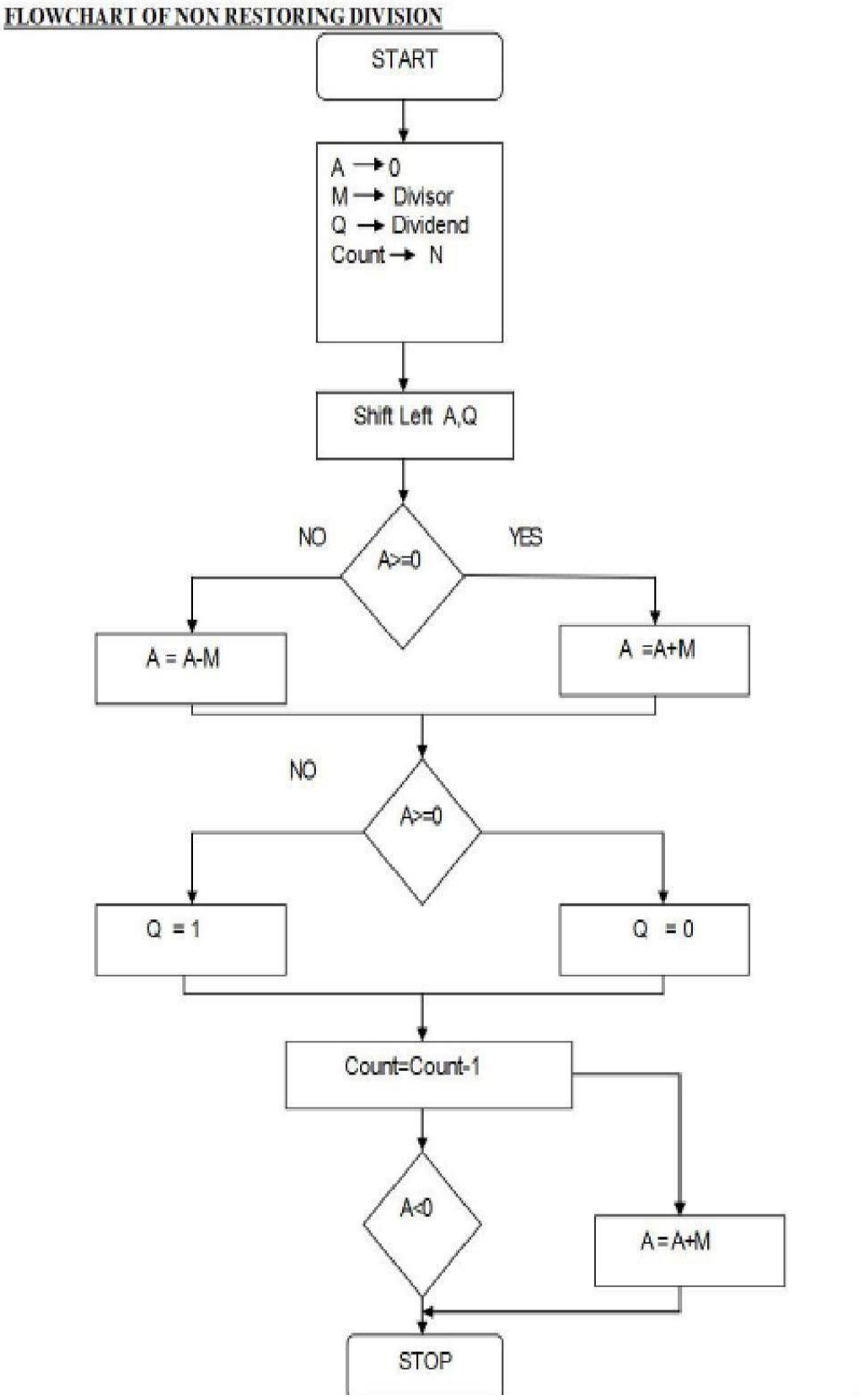
Theory:
Restoring division is usually performed on the fixed point fractional numbers. When we perform division operations on two numbers, the division algorithm will give us two things. The algorithm is based on assumption $0 < D < N$ with the help of digit set $\{0, 1\}$ the quotient digit q will be formed in the restoring division algorithm.

Algorithm:
1. Shift A and q left by 1 position.
2. Perform A ⇓ A − B
3. If sign bit of A = 1 then
   → Restore A as A ⇓ A + B
   → q ⇓ 0 = 0
4. If sign bit of A = 0
   then q ⇓ 0 = 1
5. Repeat above steps till all the bits of the dividend are used.

AIM: To write a C program for implementation of Restoring Division.

SOFTWARE: Turbo C IDE.

FLOWCHART



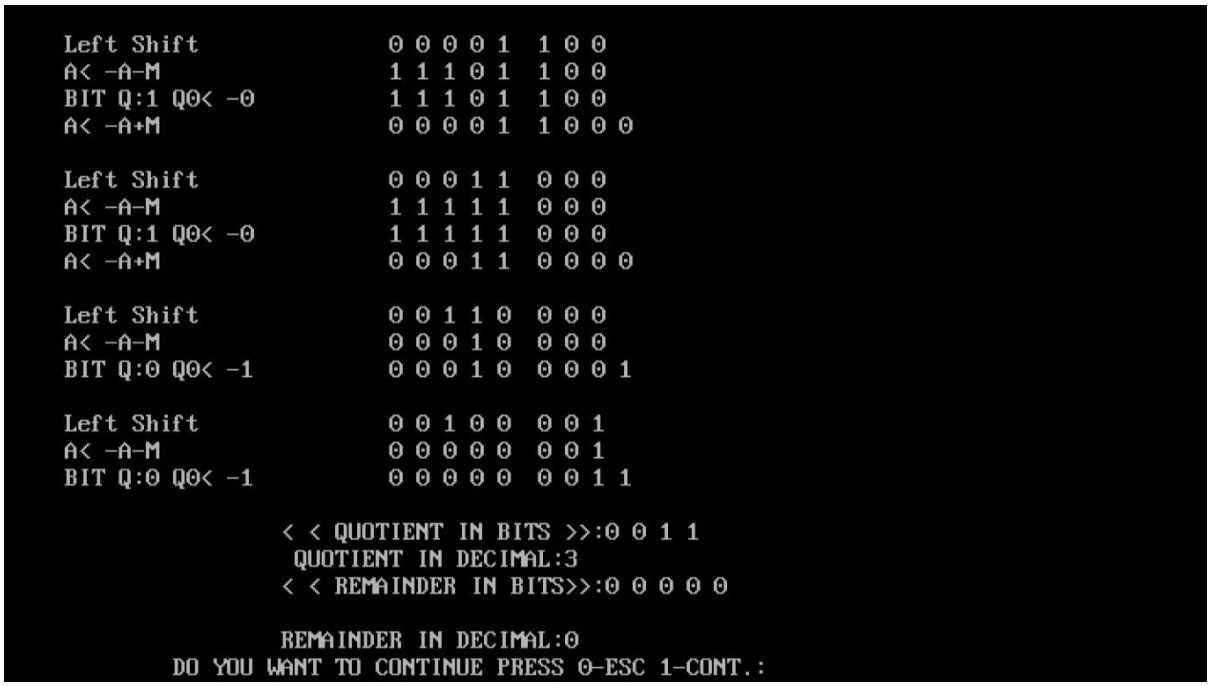
PROGRAM:

```
#include<stdio.h>
#include<math.h>
#include<conio.h>

int getsize(int x) {
int c; if(x<=1)
c = 2;
else if(x < 4) c = 2;
else if(x< 8)
c = 3;
else if(x< 16)
c = 4;
else if(x< 32)
c = 5;
else if(x< 64)
c = 6;
else if(x< 128)
c = 7;
else if(x< 256)
c = 8;
else if(x< 512)
c = 9;
return c;
}

int max(int x,int y) {
if(x< y) return(y);
else return(x);
}

void main() {
int B,Q,Z,M,c,c1,e,f,g,h,i,j,x,y,ch,in,S,G,P;
int a[24],b[12],b1[12],q[12],carry=0,count=0,option;
long num;
clrscr();
do {
printf("| |\n");
printf("|\t\tPROGRAM FOR RESTORING DIVISION\t\t|\n");
printf("| |");
printf("\n\nENTER DIVIDEND\t: ");
scanf("%d",&Q);
y = getsize(Q);
printf("ENTER DIVISOR\t: ");
scanf("%d",&M);
x = getsize(M);
Z = max(x,y);
printf("\n\tTOTAL BITS CONSIDERED FOR RESULT => %d",2*Z+1);
printf("\n\tINITiALLY A IS RESET TO ZERO:");
for(i=0;i<=Z;i++)
printf("%d ",a[i]=0);
for(i=Z;i>=0;i--) {
b1[i] = b[i] = M%2; M = M/2;
b1[i] = 1-b1[i];
}
carry = 1;
for(i=Z;i>=0;i--) {
c1 = b1[i]^carry;
```

```
carry = b1[i]&&carry;
b1[i]=c1;
}
for(i=2*Z;i>Z;i--) {
a[i] = Q%2; Q = Q/2;
}
printf("\n\n\tDivisor\t\t(M)\t: ");
for(i=0;i<=Z;i++)
printf("%d ",b[i]);
printf("\n\t2'C Divisor\t(M)\t: ");
for(i=0;i<=Z;i++)
printf("%d ",b1[i]);
printf("\n\tDividend\t(Q)\t: ");
for(i=Z+1;i<=2*Z;i++)
printf("%d ",a[i]);
printf("\n\n\tBITS CONSIDERED:[ A ][ M ]");
printf("\n\t\t\t");
for(i=0;i<=Z;i++)
printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i<=2*Z;i++)
printf("%d ",a[i]);
count = Z;
do{
for(i=0;i< 2*Z;i++)
a[i] = a[i+1];
printf("\n\nLeft Shift\t\t");
for(i=0;i<=Z;i++)
printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i< 2*Z;i++)
printf("%d ",a[i]);
carry=0;
for(i=Z;i>=0;i--) {
S=a[i]^(b1[i]^carry);
G=a[i]&&b1[i];
P=a[i]^b1[i];
carry=G||(P&&carry);
a[i]=S;
}
printf("\nA< -A-M \t\t");
for(i=0;i<=Z;i++) printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i< 2*Z;i++)
printf("%d ",a[i]);
ch=a[0];
printf("\nBIT Q:%d",ch);
switch (ch) {
case 0:
a[2*Z]=1;
printf(" Q0< -1\t\t");
for(i=0;i<=Z;i++)
printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i<=2*Z;i++)
printf("%d ",a[i]);
break;
case 1:
a[2*Z]=0;
printf(" Q0< -0\t\t");
```

```
for(i=0;i<=Z;i++)
printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i<2*Z;i++)
printf("%d ",a[i]);
carry=0;
for(i=Z;i>=0;i--) {
S=a[i]^(b[i]^carry);
G=a[i]&&b[i];
P=a[i]^b[i];
carry=G||(P&&carry);
a[i]=S ;
}
printf("\nA< -A+M");
printf("\t\t\t");
for(i=0;i<=Z;i++)
printf("%d ",a[i]);
printf(" ");
for(i=Z+1;i<=2*Z;i++)
printf("%d ",a[i]);
break;
}
count--;
} while(count!=0);
num=0;
printf("\n\n\t\t< < QUOTIENT IN BITS >>:");
for(i=Z+1;i<=2*Z;i++) {
printf("%d ",a[i]);
num=num+pow(2,2*Z-i)*a[i];
}
printf("\n\t\t QUOTIENT IN DECIMAL:%d",num);
num=0;
printf("\n\t\t< < REMAINDER IN BITS>>:");
for(i=0;i<=Z;i++) {
printf("%d ",a[i]);
num=num+pow(2,Z-i)*a[i];
}
printf("\n\n\t\tREMAINDER IN DECIMAL:%d",num);
printf("\n\tDO YOU WANT TO CONTINUE PRESS 0-ESC 1-CONT.:");
scanf("%d",&option);
} while(option!=0);
}
```

Output:



Conclusion: We learnt the Restoring division algorithm for an integer. We also learnt asimple approach to solve this problem with the help of a flow chart and applying bit operations.