

**Vidyavardhini's College of Engineering & Technology**  
Department of Artificial Intelligence and Data Science

|   |
|---|
| Experiment No. 8                                      |
| Implement Restoring algorithm using<br>c- programming |
| Name: Saurabhsing Dipaksing Pardeshi                  |
| Roll Number: 37                                       |
| Date of Performance:                                  |
| Date of Submission:                                   |

Aim: To implement Restoring division algorithm using c-programming.

Objective - 1. To understand the working of Restoring division algorithm.  
2. To understand how to implement Restoring division algorithm using c-programming.

Theory:

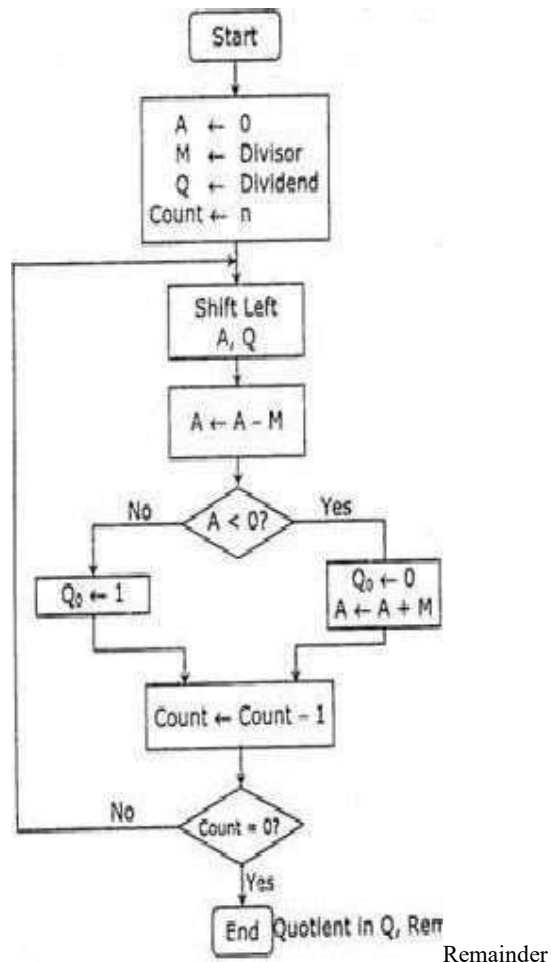
1) The divisor is placed in M register, the dividend placed in Q register. 2) At every step, the A and Q registers together are shifted to the left by 1-bit

CSL302: Digital Logic & Computer Organization Architecture Lab

3) M is subtracted from A to determine whether A divides the partial remainder. If it does, then QC) set to I-bit. Otherwise, QO gets a 0 bit and M must be added back to A to restore the previous value.

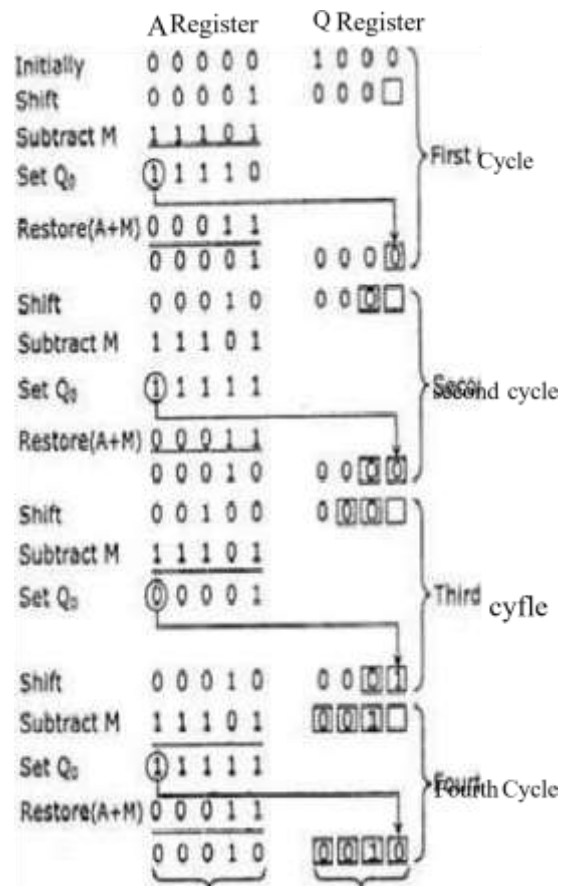
4) The count is then decremented and the process continues for n Steps. At the end, the quotient is in the Q register and the remainder is in the A register.

## Flowchart



in A

Perform • 3 by restoring division technique,



Rem"nder

Quotient

```

Program #include<stdlib.h>
#include<stdio.h>
int
acum[100]={0};
void add(int acum[],int b[],int n); int
q[100],b[100];
int main()
{
    int x,y;
    printf("Enter the Number :");
    scanf("%d%d", &x,&y);
  
```

```

    int i=0;
while(x>0 | y>0)
    {
    if(x>0)
        {
            q[i]=x%2;
x=x/2;
        }
    else {
q[i]=0;
}
        if(y>0)
        {
            b[i]=y%2;
y=y/2;
        }
    else {
b[i]=0;
}        i++;
    }

```

```

int n=i;
int bc[50];
printf("\n");
for(i=0;i<n;i++)
    {
        if(b[i]==0)
        {
            bc[i]=1;
        }
    else
        {
            bc[i]=0;
        }
    }
    bc[n]=1;
for(i=0;i<=n;i++)
    {
        if(bc[i]==0)
        {
            bc[i]=1;
i=n+2;
        }
    else
    {
        bc[i]=0;
    } } int l;
b[n]=0;    int

```

```

k=n;  int
n1=n+n-1;
int j,mi=n-1;
    for(i=n;i!=0;i--)
    {
        for(j=n;j>0;j--)
        {
            acum[j]=acum[j-1];
        }
        acum[0]=q[n-1];
        for(j=n-1;j>0;j--)
        {
            q[j]=q[j-1];
        }
        add(acum,bc,n+1);
        if(acum[n]==1)
        {
            q[0]=0;
add(acum,b,n+1);
        }
    else
    {
        q[0]=1;
    }
    }
    printf("\nQuoient : ");

    for( l=n1;l>=0;l--)
    {
        printf("%d",q[l]);
    }
    printf("\nRemainder : ");
    for( l=n;l>=0;l--)
    {
        printf("%d",acum[l]);
    }
    return 0;
}

void add(int acum[],int bo[],int n)
{
    int
    i=0,temp=0,sum =0;
    for(i=0;i<n;i++)
    {
        sum=0;
        sum=acum[i]+bo[i]+temp;
        if(sum==0)
        {

```

```
        acum[i]=0;
temp=0;
    }
    else
if (sum==2)
    {
        acum[i]=0;
temp=1;
    }
else
if(sum==1)
    {
        acum[i]=1;
temp=0;
    }
else
    if(sum==3)
    {
        acum[i]=1;
temp=1;
    }
    }
}
```

Output-

```

Enter the Dividend: 15
Enter the Divisor: 5
A   Q   Comments
0000 1111 Start
0001 111_ Left Shift A,Q
1100 111_ A=A-M
0001 1110 Qo=0; A=A+M
0011 110_ Left Shift A,Q
1110 110_ A=A-M
0011 1100 Qo=0; A=A+M
0111 100_ Left Shift A,Q
0010 100_ A=A-M
0010 1001 Qo=1
0101 001_ Left Shift A,Q
0000 001_ A=A-M
0000 0011 Qo=1

Quotient = 0011 Remainder = 0000

```

Conclusion - The aim of the experiment is to implement the Restoring division algorithm in C programming, a method for efficiently performing division by restoring partial remainders and quotients, aiming to optimize the division process and achieve accurate results in a systematic manner.