**Name:Gaurav Kishor Patil**

**Roll No:54 Batch:C**

**Div:2**

|  |
| --- |
| Experiment No. 7 |
| Implement Restoring algorithm using c-programming |
| Date of Performance:13/09/23 |
| Date of Submission:27/09/23 |

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

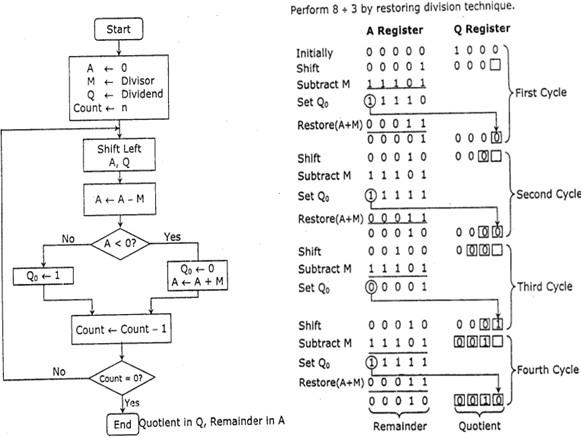
# Objective -

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
3. M is subtracted from A to determine whether A divides the partial remainder. If it does, then Q0 set to 1-bit. Otherwise, Q0 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



**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=n-1;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;

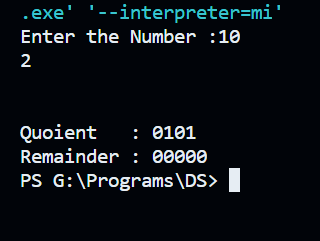
temp=1;

}

}

}

**Output –**

****

**Conclusion -**

Restoring division algorithm is used to divide two unsigned integers. It operates on fixed-point fractional numbers and depends on the following assumptions: The following division methods are all based on the form Q = A/ M where Q = Quotient A = Numerator (dividend) M = Denominator (divisor). The algorithm starts with Quotient = 0, Remainder =0 and Sign = 0. It then asks the user to enter two decimal numbers: n1, n2. The program converts their absolute values into binary and stores them in arrays num1 and num2. Two’s complement num2 and store as ncom. Create a copy of num1 as ncopy. If the product is negative, set sign = 1. Shift left Remainder : ncopy; counter = 0. Add ncom to Remainder. Set LSB of ncopy as 0. If result is negative, restore the remainder. Otherwise, Set LSB of ncopy as 1. If counter < bits in num1, Shift left Remainder : ncopy counter = counter + 1. Repeat steps 9, 10 and 11 until all bits of num1 is traced. Finally Display the result