**Batch: B2 Roll No.: 1611103**

**Experiment / assignment / tutorial No. 3**

**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 involve 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:**

CO 2-Detail working of the arithmetic logic unit and its sub modules

CO 3-Understand the Central processing unit with addressing modes and working of control unit

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

**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.

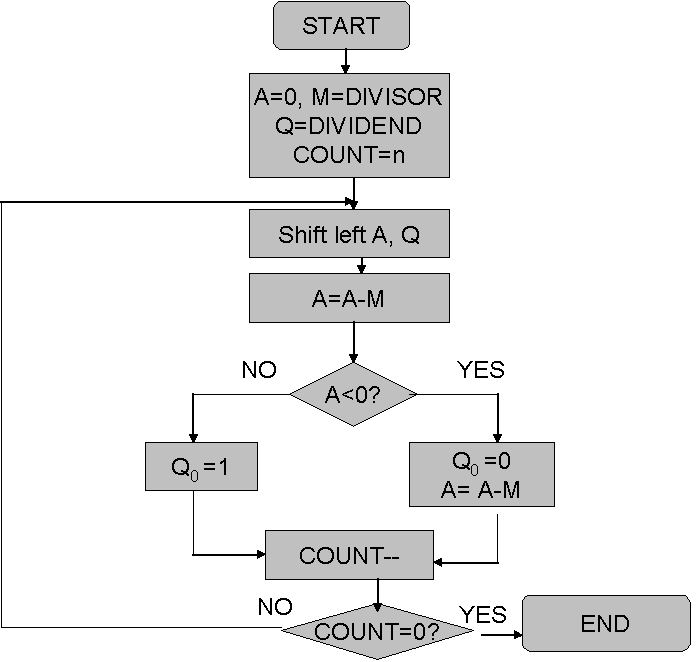
**3**. Dr. M. Usha, T. S. Srikanth, “Computer System Architecture and Organization”, First Edition, Wiley-India.

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

**Pre Lab/ Prior Concepts:**

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

**Flowchart for Restoring of Division:**



**Design Steps**:

1. 1Start
2. 2Initialize 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:**

**#include<stdio.h>**

**#include<conio.h>**

**#define MAXB 15**

**#define MAX 16**

**//using namespace std;**

**void bic(int l,int p[])**

**{**

**int i = 0;**

**while(l != 0)**

**{**

**p[i] = l%2;**

**i++;**

**if(i >= MAX)**

**l = 0;**

**else**

**l = l/2;**

**}**

**for(;i<MAX;i++)**

**{**

**p[i] = 0;**

**}**

**}**

**void printb(int n[])**

**{**

**for(int i = MAXB;i>=0;i--)**

**{**

**printf("%d",n[i]);**

**}**

**printf("\t");**

**}**

**int check(int a[])**

**{**

**if(a[MAXB] == 1)**

**return 1;**

**else if(a[MAXB] == 0)**

**return 2;**

**else**

**return 3;**

**}**

**void add(int A[],int m[])**

**{**

**int carry = 0;**

**for(int i =0;i<MAX;i++)**

**{**

**A[i] = A[i] + m[i] + carry ;**

**carry = 0;**

**if(A[i] == 2)**

**{**

**A[i] = 0;**

**carry = 1;**

**}**

**else if(A[i] == 3)**

**{**

**A[i] = 1;**

**carry = 1;**

**}**

**else**

**continue;**

**}**

**}**

**void sub(int a[],int m[])**

**{**

**int n[MAX];**

**for(int j = 0;j<MAX;j++)**

**{**

**n[j] = m[j];**

**}**

**for(int i = MAXB;i>=0;i--)**

**{**

**if(n[i] == 0)**

**{**

**n[i] = 1;**

**continue;**

**}**

**else if(n[i] == 1)**

**{**

**n[i] = 0;**

**continue;**

**}**

**}**

**int m1[] = {1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};**

**add(n,m1);**

**add(a,n);**

**}**

**void shift(int a[],int q[])**

**{**

**int temp = q[MAXB];**

**for(int i = MAXB;i>0;i--)**

**{**

**a[i] = a[i-1];**

**q[i] = q[i-1];**

**}**

**a[0] = temp;**

**}**

**void main()**

**{**

**clrscr();**

**int M,Q,count = MAX;**

**printf("Enter the numerator : ");**

**scanf("%d",&Q);**

**printf("Enter the denominator : ");**

**scanf("%d",&M);**

**int m[MAX],q[MAX];**

**bic(M,m);**

**bic(Q,q);**

**//printb(m);**

**//printb(q);**

**int A[16];**

**for(int i = 0;i<MAX;i++)**

**{**

**A[i] = 0;**

**}**

**label:**

**shift(A,q);**

**sub(A,m);**

**switch(check(A))**

**{**

**case 1:**

**q[0] = 0;**

**add(A,m);**

**break;**

**case 2:**

**q[0] = 1;**

**break;**

**case 3:printf("Error");**

**count = 0;**

**break;**

**}**

**count--;**

**if(count>0)**

**goto label;**

**else**

**{**

**printf("The division of the given nos: \nRemainder: ");**

**printb(A);**

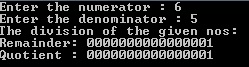
**printf("\nQuotient : ");**

**printb(q);**

**}**

**//return 0;**

**getch();**

**}**

**Conclusion:**

**The program for restoring division has been successfully executed.**

**Post Lab Descriptive Questions (Add questions from examination point view)**

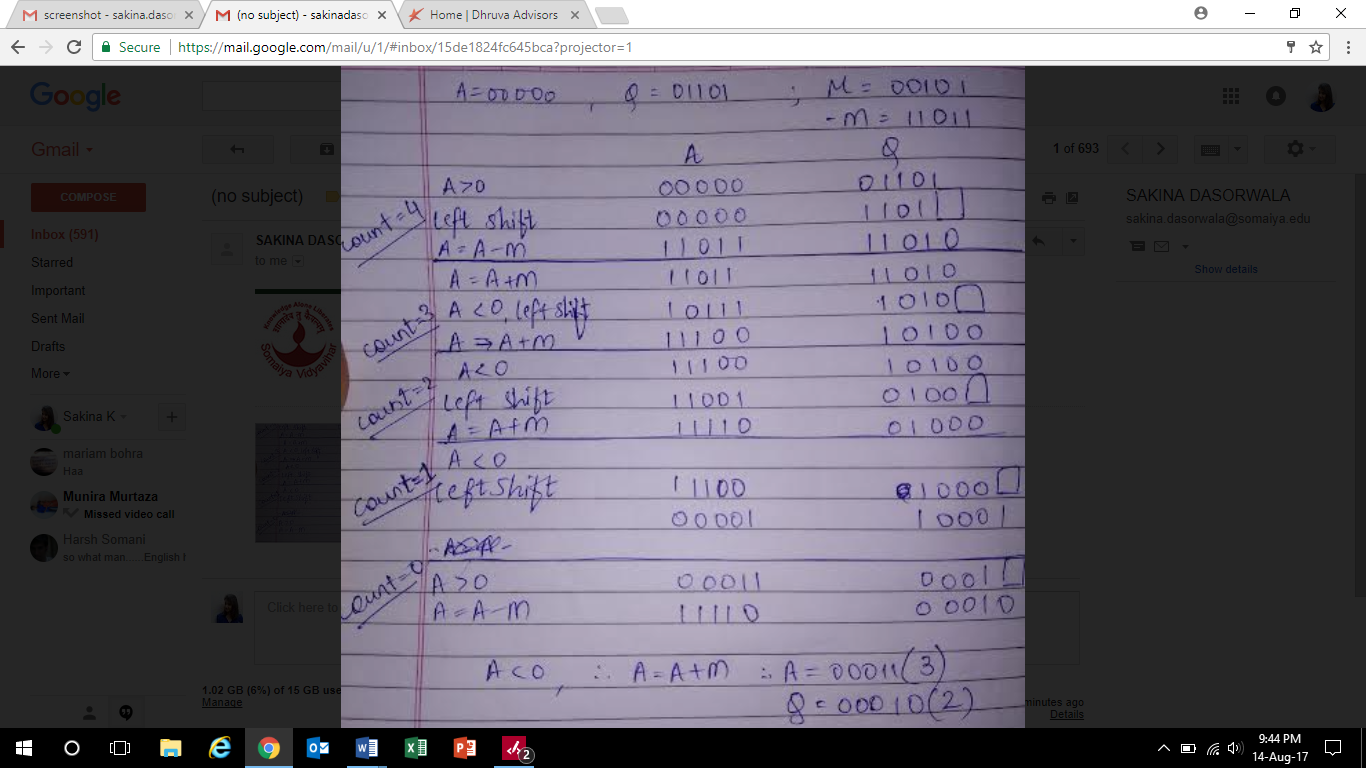
1. **What are the advantages of non-restoring division over restoring division?**

**2. Simulate non restoring division algorithm for unsigned numbers A=1101 and B=0101**

**Ans 1)**

Non restoring dividion prevents the need for an extra restoring operation in the restoring step and thus reduces the step computation.

**Ans 2)**



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