K. J. Somaiya College of Engineering, Mumbai-77

Batch: B1 Roll No.: 1711072

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

---

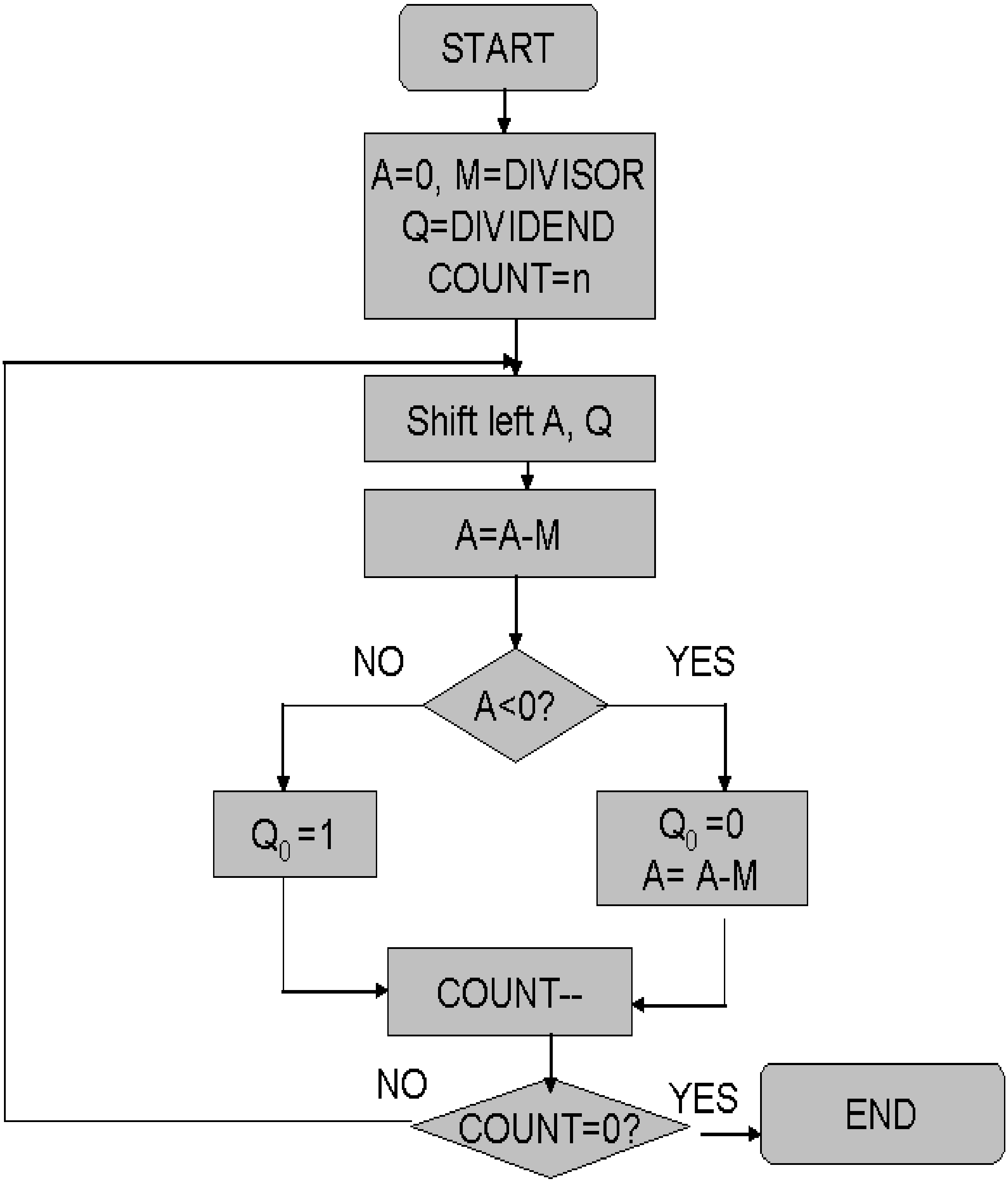
Books/ Journals/ Websites referred:

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. 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;
    - Then A=A-M
    - Else A=A+M
5. If MSB of previous A and present A are same
    - Q_0=0 & store present A
    - Else Q_0=0 & restore previous A
6. Decrement count.
7. If count=0, go to 11, else go to 3.
8. STOP

Implementation (in Java):

```
import java.util.*;
class Main {
  public static void main(String[] args) {
    int M,Q;
    int BITS=5;
    int count=BITS;
    int M_BIN[]=new int[BITS];
    int Q_BIN[]=new int[BITS];
    int answer[]=new int[BITS];
    int M_TEMP[]=new int[BITS];
    Scanner sc=new Scanner(System.in);
    System.out.println("RESTORING DIVISION: ");
    System.out.print("Enter Divisor(M): ");
    M=sc.nextInt();
    System.out.print("Enter Dividend(Q): ");
    Q=sc.nextInt();

    M_BIN=binary(M, BITS);
    Q_BIN=binary(Q, BITS);
    if(M<0){
      M_BIN=complement(M_BIN, BITS);
    }
    if(Q<0){
```

```
    Q_BIN=complement(Q_BIN, BITS);
  }
  System.out.print("\nM in binary is: ");
  Arrays.stream(M_BIN).forEach(System.out::print);
  System.out.print("\nQ in binary is: ");
  Arrays.stream(Q_BIN).forEach(System.out::print);
  System.out.println("\n\nExpected product: "+(M*Q));
  Arrays.stream(answer).forEach(System.out::print);
  System.out.print(" : ");
  Arrays.stream(Q_BIN).forEach(System.out::print);
  while(count>0){
  System.out.println("\n\nCOUNT= "+count);
  for(int i=0;i<BITS-1;i++){
    answer[i]=answer[i+1];
  }
  answer[BITS-1]=Q_BIN[0];
  for(int i=0;i<BITS-1;i++){
    Q_BIN[i]=Q_BIN[i+1];
  }
  System.out.println("\nArray after left shift and A=A-M: ");
  M_TEMP=complement(M_BIN.clone(), BITS);
  answer=add(answer, M_TEMP, BITS);
  Arrays.stream(answer).forEach(System.out::print);
  System.out.print(" : ");
  Arrays.stream(Q_BIN).forEach(System.out::print);
  if(answer[0]==0){
      Q_BIN[BITS-1]=1;
      System.out.println("\nArray after Q0=1: ");
      Arrays.stream(answer).forEach(System.out::print);
      System.out.print(" : ");
      Arrays.stream(Q_BIN).forEach(System.out::print);
  }

  else if(answer[0]==1){
    Q_BIN[BITS-1]=0;
    M_TEMP=M_BIN.clone();
    answer=add(answer, M_TEMP, BITS);
    System.out.println("\nArray after Q0=0 & A=A+M: ");
    Arrays.stream(answer).forEach(System.out::print);
    System.out.print(" : ");
    Arrays.stream(Q_BIN).forEach(System.out::print);
  }
  count--;
}
```

```
    System.out.print("\nRemainder : Quotient:  ");
    Arrays.stream(answer).forEach(System.out::print);
    System.out.print(" : ");
    Arrays.stream(Q_BIN).forEach(System.out::print);
  }
  public static int[] binary(int dec, int BITS){
    int[] dec_bin=new int[BITS];
    for(int i=BITS-1;i>=0;i--){
      dec_bin[i]=dec%2;
      dec=dec/2;
    }
    return dec_bin;
  }
  public static int[] add(int[] arr1,int[] arr2, int BITS){
    int carry=0;
    int arr[]=new int[BITS];
    for(int i=BITS-1;i>=0;i--)
    {
        arr[i]=(arr1[i]+arr2[i]+carry)%2;
        carry=(arr1[i]+arr2[i]+carry)/2;
    }
    return arr;
  }
  public static int[] complement(int[] bin, int BITS){
    for(int i=0;i<BITS;i++){
        bin[i]=(bin[i]==0)?1:0;
    }
    int plus_one[]=new int[BITS];
    plus_one[BITS-1]=1;
    bin=add(bin,plus_one,BITS);
    return bin;
  }
}
```

For verification, my code is available on:
https://repl.it/@ARGHYADEEPDAS/COAExpt3

Output Screen:

```
RESTORING DIVISION:
Enter Divisor(M):  11
Enter Dividend(Q):  25

M in binary is: 01011
Q in binary is: 11001

00000 : 11001

COUNT= 5

Array after left shift and A=A-M:
10110 : 10011
Array after Q0=0 & A=A+M:
00001 : 10010

COUNT= 4

Array after left shift and A=A-M:
11000 : 00100
Array after Q0=0 & A=A+M:
00011 : 00100

COUNT= 3

Array after left shift and A=A-M:
11011 : 01000
Array after Q0=0 & A=A+M:
00110 : 01000
COUNT= 2

Array after left shift and A=A-M:
00001 : 10000
Array after Q0=1:
00001 : 10001

COUNT= 1

Array after left shift and A=A-M:
11000 : 00011
Array after Q0=0 & A=A+M:
00011 : 00010
Remainder : Quotient:  00011 : 00010>
```

Example:
M=0101; Q=1101

Count	A	Q	Operation
4	0000	1101	
	0001	101_	Arithmetic Left Shift
	1100	101_	A=A-M
	0001	1010	Q_0=0; A=A+M
3	0011	010_	Arithmetic Left Shift
	1110	010_	A=A-M
	0011	0100	Q_0=0; A=A+M
2	0110	100_	Arithmetic Left Shift
	0001	1001	A=A-M; Q_0=1
1	0011	001_	Arithmetic Left Shift
	1110	001_	A=A-M
	0011	**0010**	Q_0=0; A=A+M

Remainder=A=0011; Quotient=Q=0010.

Conclusion:

Restoring division algorithm was successfully executed and hence verified as the desired outputs were achieved.

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

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

Ans.

1. Non-restoring division is faster as it requires less time; whereas restoring division is slower as there is restoration in each cycle.
2. Sign bit determines whether addition or subtraction is to be performed in non-restoring, whereas there is no such thing in restoring division.

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

```
M in binary is: 0101
Q in binary is: 1101

0000 : 1101

COUNT= 4

Array after left shift and A=A-M:
1100 : 1011
Array after Q0=0 & A=A+M:
0001 : 1010

COUNT= 3

Array after left shift and A=A-M:
1110 : 0100
Array after Q0=0 & A=A+M:
0011 : 0100

COUNT= 2

Array after left shift and A=A-M:
0001 : 1000
Array after Q0=1:
0001 : 1001

COUNT= 1

Array after left shift and A=A-M:
1110 : 0011
Array after Q0=0 & A=A+M:
0011 : 0010
Remainder : Quotient:  0011 : 0010>
```

Date: 01/08/2018 **Signature of faculty in-charge**