Jawahar Education Societys Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

NAME: PRIYUSH BHIMRAO KHOBRAGADE

PRN NO: 211112018

SUBJECT: DIGITAL LOGIC & COMPUTER ORGANIZATION AND ARCHITECTURE LAB

04

AIM: To implement non restoring division algorithm.

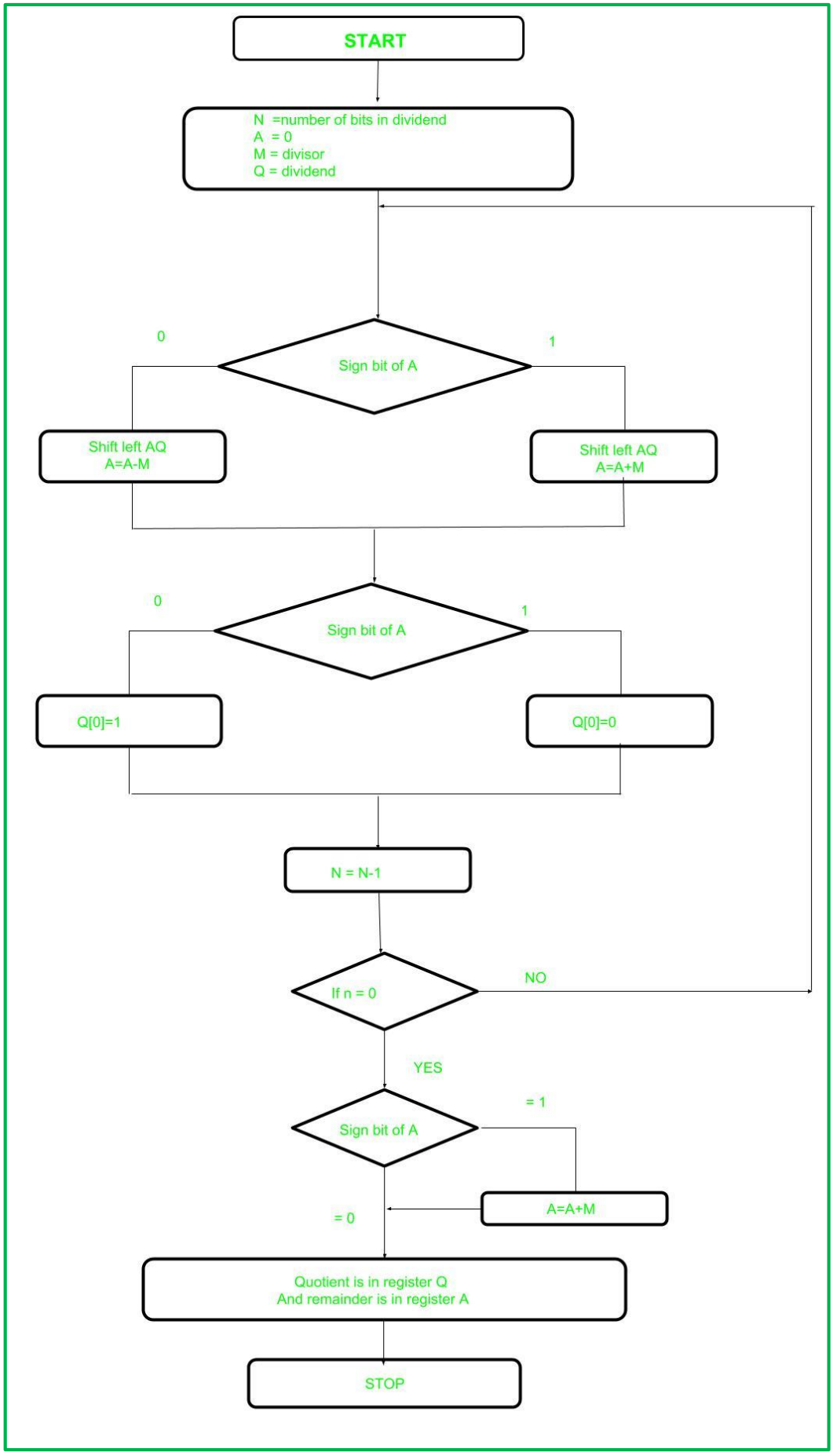
Practical No.4

- **Aim**: To implement non restoring division algorithm.
- **Objectives**: To implement the operation of the arithmetic unit using the non-restoring division algorithm.
- **Outcomes**: Learner will able to understand the implementation and working of non-restoring division algorithm.
- **Hardware / Software Required:** Any programming language C, Java etc.
- **Theory**:

Non-Restoring Division Algorithm:

In earlier post Restoring Division learned about restoring division. Now, here perform Non-Restoring division, it is less complex than the restoring one because simpler operation are involved i.e. addition and subtraction, also now restoring step is performed. In the method, rely on the sign bit of the register which initially contain zero named as A.

AIM: To implement restoring division algorithm



AIM: To implement restoring division algorithm

Let's pick the step involved:

- **Step-1:** First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)
- **Step-2:** Check the sign bit of register A
- **Step-3:** If it is 1 shift left content of AQ and perform A = A+M, otherwise shift left AQ and perform A = A-M (means add 2's complement of M to A and store it to A)
- **Step-4:** Again the sign bit of register A
- **Step-5:** If sign bit is 1 Q[0] become 0 otherwise Q[0] become 1 (Q[0] means least significant bit of register Q)
- **Step-6:** Decrements value of N by 1
- **Step-7:** If N is not equal to zero go to **Step 2** otherwise go to next step
- **Step-8:** If sign bit of A is 1 then perform A = A+M
- **Step-9:** Register Q contain quotient and A contain remainder

Program Input:

```
import java.util.*;


class nonresto
{
 public static int n,c;
 public static int A[]=new int[20];
 public static int Q[]=new int [20];
 public static int M[]=new int [20];
 public static int M1[]=new int[20];


 public static void main(String args[])
 {
  int x,y,cin,temp,i,count,j=0;
  Scanner sc=new Scanner(System.in);
  System.out.print("\nEnter no. of bits: ");
  n=sc.nextInt();
```

AIM: To implement restoring division algorithm

```
System.out.print("Enter dividend: ");
y=sc.nextInt();
 Q=toB(y,n);
System.out.print("Enter divisor: ");
x=sc.nextInt();
 M=toB(x,n+1);
 M1=toB(x,n+1);
 M1=twoC(M1,n+1);
A=toB(0,n);
c=0;
System.out.print("\nDividend: ");
for(i=n-1;i>=0;i--)
 System.out.print(Q[i]);
System.out.print("\nDivisor: ");
for(i=n;i>=0;i--)
 System.out.print(M[i]);
count=n;

while(count>0)
{
 if(c==0)
 {
  lshift();
  cin=0;
  for(i=0;i<n;i++)
  {
   temp=A[i];
   A[i]=(A[i]+M1[i]+cin)%2;
   cin=(temp+M1[i]+cin)/2;
```

```
 }
 c=(c+cin+M1[i])%2;
}
else
{
 lshift();
 cin=0;
 for(i=0;i<n;i++)
 {
  temp=A[i];
  A[i]=(A[i]+M[i]+cin)%2;
  cin=(temp+M[i]+cin)/2;
 }
 c=(c+cin+M[i])%2;
}
if(c==0)
 Q[0]=1;
else
 Q[0]=0;
j++;
System.out.print("\nCycle "+j+": ");
for(i=n-1;i>=0;i--)
 System.out.print(A[i]);
for(i=n-1;i>=0;i--)
 System.out.print(Q[i]);
count--;
}

System.out.print("\n\nResult: ");
```

AIM: To implement restoring division algorithm

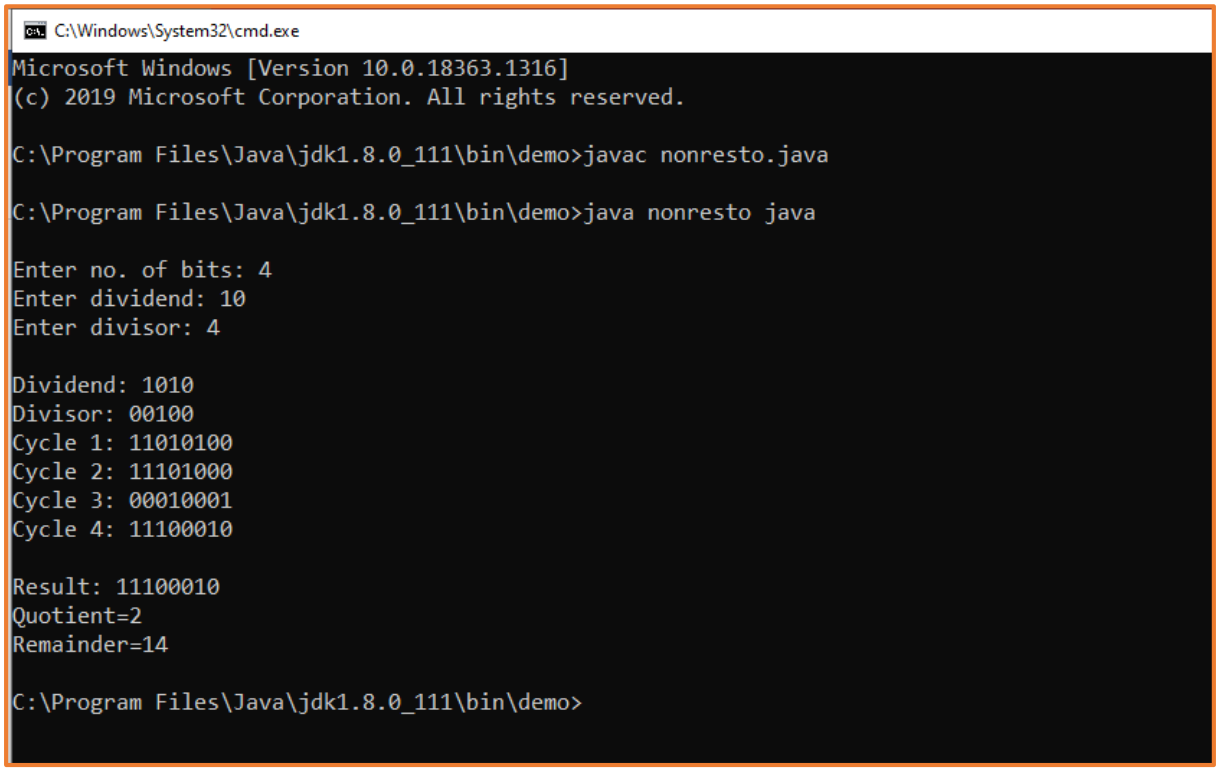
```
 for(i=n-1;i>=0;i--)
  System.out.print(A[i]);
 for(i=n-1;i>=0;i--)
  System.out.print(Q[i]);
 double s=toD(Q);
 System.out.println("\nQuotient="+(int)s);
 s=toD(A);
 System.out.println("Remainder="+(int)s);
}

public static int[] toB(int x,int n)
{
 int i=0;
 int t[]=new int[n];
 while(x>0&&i<n)
 {
  t[i]=x%2;
  x=x/2;
  i++;
 }
 if(x>0)
 {
   System.out.println("Overflow!\nSize of register is insufficient!");
   System.exit(0);
  }
 while(i<=n-1)
 {
  t[i]=0;
  i++;
```

```
 }
 return t;
}

public static double toD(int a[])
{
 int i=0;
 double s=0;
 while(i<n)
 {
   s=s+(a[i]*Math.pow(2,i));
   i++;
 }
 return(s);
}

public static int[] twoC(int a[], int m)
{
 int i=0;
 for(i=0;i<m;i++)
  a[i]=(a[i]+1)%2;
 i=0;
 while(a[i]!=0&&i<n)
 {
  a[i]=0;
  i++;
 }
 a[i]=1;
 return a;
```

```
 }

 public static void lshift()
 {
  int i;
  c=A[n-1];
  for(i=n-1;i>0;i--)
   A[i]=A[i-1];
  A[0]=Q[n-1];
  for(i=n-1;i>0;i--)
   Q[i]=Q[i-1];
 }
}
```

AIM: To implement restoring division algorithm

Output:



● **Conclusion:** Hence, It is to be concluded that this presentation deals with the design **approach of non-restoring division algorithm**.