**Experiment No: 02**

**Aim: To implement Restoring and Non-restoring method for integer division.**

**Code and Output:**

**Restoring Method:**

def shift\_left(s):

s=s[1:]

s=s+"0"

return (s)

def complement(s):

d={'0':'1','1':'0'}

e=''.join(d[x] for x in s)

l=len(e)

sum=bin(int(e,2)+int("1",2))

sum=sum[2:]

return(sum.zfill(l))

def addzerotostring(x):

x=list(x)

x[-1]="0"

return("".join(x))

def addonetostring(x):

x=list(x)

x[-1]="1"

return("".join(x))

q=int(input("Enter the dividend :"))

m=int(input("Enter the divisor :"))

Q=bin(q)

M=bin(m)

# print('q',Q)

Q=Q[2:] #final Q

M=M[2:] #final M

# print('q',Q)

if(len(M)>len(Q)):

Q=Q.zfill(len(M))

else:

M=M.zfill(len(Q))

print("Binary Value Of M :",M)

print("Binary Value Of Q :",Q)

l=len(Q)

count=l

Mc=complement(M)

a="0"

for i in range(0,l-1):

a=a+"0"

while(count>0):

s=a+Q

value=shift\_left(s)

a=value[0:l]

Q=value[l:]

a=bin(int(a,2)+int(Mc,2))

if(len(a[2:])==l+1):

a=a[3:]

else:

a=a[2:]

if(a[0]=="1"):

a=bin(int(a,2)+int(M,2))

if(len(a[2:])==l+1):

a=a[3:]

else:

a=a[2:]

Q=addzerotostring(Q)

else:

Q=addonetostring(Q)

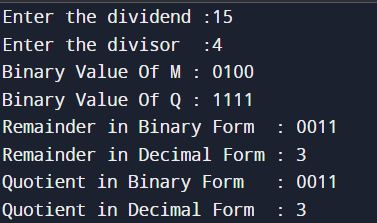
count=count-1

print("Remainder in Binary Form :",a)

print("Remainder in Decimal Form :",int(a,2))

print("Quotient in Binary Form :",Q)

print("Quotient in Decimal Form :",int(Q,2))

****

**Non-Restoring Method:**

def add(A, M):

carry = 0

Sum = ''

for i in range (len(A)-1, -1, -1):

temp = int(A[i]) + int(M[i]) + carry

if (temp>1):

Sum += str(temp % 2)

carry = 1

else:

Sum += str(temp)

carry = 0

return Sum[::-1]

def compliment(m):

M = ''

for i in range (0, len(m)):

M += str((int(m[i]) + 1) % 2)

M = add(M, '0001')

return M

def nonRestoringDivision(Q, M, A):

count = len(M)

comp\_M = compliment(M)

flag = 'successful'

print ('Initial Values: A:', A,

' Q:', Q, ' M:', M)

while (count):

print ("\nstep:", len(M)-count + 1,

end = '')

print (' Left Shift and ', end = '')

A = A[1:] + Q[0]

if (flag == 'successful'):

A = add(A, comp\_M)

print ('subtract: ')

else:

A = add(A, M)

print ('Addition: ')

print('A:', A, ' Q:',

Q[1:]+'\_', end ='')

if (A[0] == '1'):

Q = Q[1:] + '0'

print (' -Unsuccessful')

flag = 'unsuccessful'

print ('A:', A, ' Q:', Q,

' -Addition in next Step')

else:

Q = Q[1:] + '1'

print (' Successful')

flag = 'successful'

print ('A:', A, ' Q:', Q,

' -Subtraction in next step')

count -= 1

print ('\nQuotient(Q):', Q,

' Remainder(A):', A)

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

dividend = input('Enter 4 digit binary dividend:')

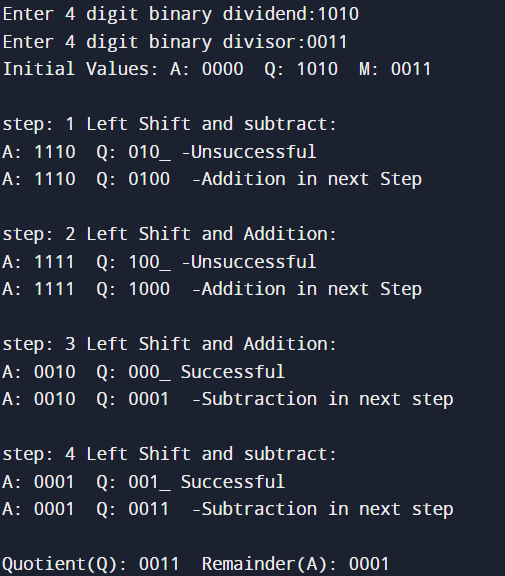
divisor =input('Enter 4 digit binary divisor:')

accumulator = '0' \* len(dividend)

nonRestoringDivision(dividend,

divisor,

accumulator)

****