Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**A.Y. 2022-2023**

# PROCESSOR ORGANIZATION AND ARCHITECTURE

## AYUSH JAIN

**COMPUTER ENGINEERING | TE – B2 | 60004200132**

## EXPERIMENT – 2

**AIM:** To study and implement Restoring division algorithm and Non-Restoring division algorithm

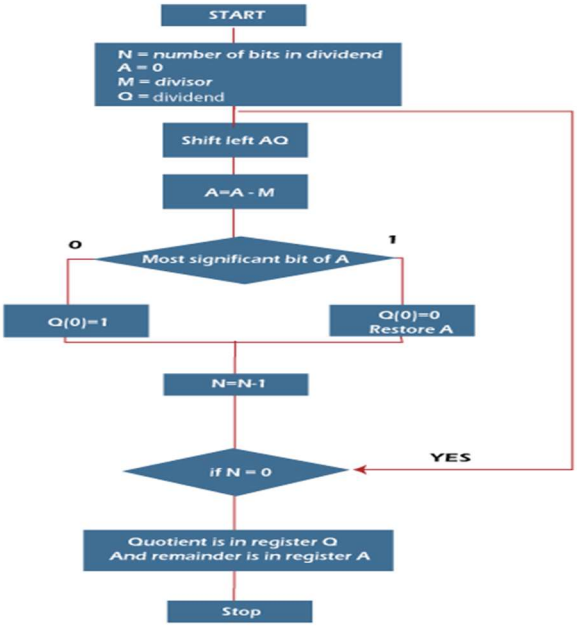
**THEORY:**

Restoring division: Restoring division is usually performed on the fixed-point fractional numbers. When we perform division operations on two numbers, the division algorithm will give us two things, i.e., quotient and remainder. This algorithm is based on the assumption that $0 < D < N$. With the help of digit set {0, 1}, the quotient digit q will be formed in the restoring division algorithm.

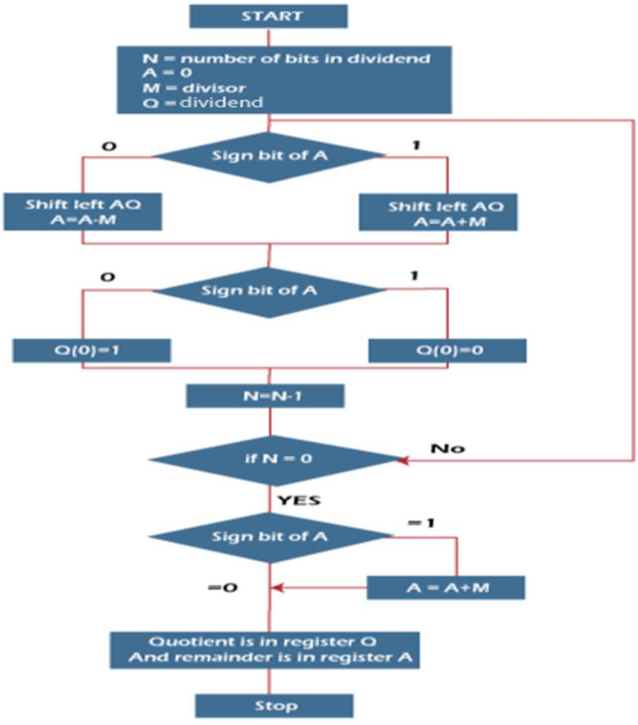
Non-Restoring division: Instead of the quotient digit set {0, 1}, the set {-1, 1} is used by the non-restoring division. The non-restoring division algorithm is more complex as compared to the restoring division algorithm. But when we implement this algorithm in hardware, it has an advantage, i.e., it contains only one decision and addition/subtraction per quotient bit. After performing the subtraction operation, there will not be any restoring steps. Due to this, the numbers of operations basically cut down up to half. Because of the less operation, the execution of this algorithm will be fast. This algorithm basically performs simple operations such as addition, subtraction. In this method, we will use the sign bit of register A. 0 is the starting value/bit of register A.

## FLOW CHART:

### Restoring division:




### Non-Restoring division:



**ALGORITHM:**

**Restoring Division:**

**Step 1:** In this step, the corresponding value will be initialized to the registers, i.e., register A will contain value 0, register M will contain Divisor, register Q will contain Dividend, and N is used to specify the number of bits in dividend.

**Step 2:** In this step, register A and register Q will be treated as a single unit, and the value of both the registers will be shifted left.

**Step 3:** After that, the value of register M will be subtracted from register A. The result of subtraction will be stored in register A.

**Step 4:** Now, check the most significant bit of register A. If this bit of register A is 0, then the least significant bit of register Q will be set with a value 1. If the most significant bit of A is 1, then the least significant bit of register Q will be set to with value 0, and restore the value of A that means it will restore the value of register A before subtraction with M.

**Step 5:** After that, the value of N will be decremented. Here n is used as a counter.

**Step 6:** Now, if the value of N is 0, we will break the loop. Otherwise, we have to again go to step 2.

**Step 7:** This is the last step. In this step, the quotient is contained in the register Q, and the remainder is contained in register A.

**Non-Restoring division:**

**Step 1:** In this step, the corresponding value will be initialized to the registers, i.e., register A will contain value 0, register M will contain Divisor, register Q will contain Dividend, and N is used to specify the number of bits in dividend.

**Step 2:** In this step, we will check the sign bit of A.

**Step 3:** If this bit of register A is 1, then shift the value of AQ through left, and perform A = A + M. If this bit is 0, then shift the value of AQ into left and perform

A = A - M. That means in case of 0, the 2's complement of M is added into register A, and the result is stored into A.

**Step 4:** Now, we will check the sign bit of A again.

**Step 5:** If this bit of register A is 1, then Q[0] will become 0. If this bit is 0, then Q[0] will become 1. Here Q[0] indicates the least significant bit of Q.

**Step 6:** After that, the value of N will be decremented. Here N is used as a counter.

**Step 7:** If the value of N = 0, then we will go to the next step. Otherwise, we have to again go to step 2.

**Step 8:** We will perform A = A + M if the sign bit of register A is 1.

**Step 9:** This is the last step. In this step, register A contains the remainder, and register Q contains the quotient.

**CODE:**

**Restoring Division:**

```
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 restoringDivision(Q, M, A):
  count = len(M)
```

```
  print ('Initial Values: A: ', A,' Q: ', Q, ' M: ', M)
  while (count):
    print ("\nstep:", len(M)-count + 1, end = '')
    print (' Left Shift and Subtract: \n')
    A = A[1:] + Q[0]
    comp_M = compliment(M)
    A = add(A, comp_M)
    print('A: ', A, ' Q: ', Q[1:]+'_', end ='')
    if (A[0] == '1'):
      Q = Q[1:] + '0'
      print (' - Unsuccessful')
      A = add(A, M)
      print ('A: ', A, ' Q: ', Q, ' - Restoration')
    else:
      Q = Q[1:] + '1'
      print (' - Successful')
      print ('A: ', A, ' Q: ',
        Q, ' - No Restoration')
    count -= 1
  print ('\nQuotient(Q): ', Q, ' Remainder(A): ', A)

dividend = input("Enter the dividend(Q) in binary: ")
divisor = input("Enter the divisor(M) in binary: ")
accumulator = '0' * len(dividend)
restoringDivision(dividend, divisor, accumulator)
```

**OUTPUT:**

```
Initial Values: A: 0000  Q: 0110  M: 0100

step: 1 Left Shift and Subtract:  A: 1100
A: 1100  Q: 110_  -Unsuccessful
A: 0000  Q: 1100  -Restoration

step: 2 Left Shift and Subtract:  A: 1101
A: 1101  Q: 100_  -Unsuccessful
A: 0001  Q: 1000  -Restoration

step: 3 Left Shift and Subtract:  A: 1111
A: 1111  Q: 000_  -Unsuccessful
A: 0011  Q: 0000  -Restoration

step: 4 Left Shift and Subtract:  A: 0010
A: 0010  Q: 000_ Successful
A: 0010  Q: 0001  -No Restoration

Quotient(Q): 0001  Remainder(A): 0010
PS C:\.vscode\college>
```

## CODE:

## Non-Restoring Division:

```
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')
```

```
46.         count -= 1
47.     print ('\nQuotient(Q):', Q,' Remainder(A):', A)
48. dividend = input("Enter the dividend(Q) in binary: ")
49. divisor = input("Enter the divisor(M) in binary: ")
50. accumulator = '0' * len(dividend)
51. nonRestoringDivision(dividend, divisor, accumulator)
```

**OUTPUT:**

```
Enter the dividend(Q) in binary: 0110
Enter the divisor(M) in binary: 0100
Initial Values: A: 0000  Q: 0110  M: 0100

step: 1 Left Shift and subtract:
A: 1100  Q: 110_  -Unsuccessful
A: 1100  Q: 1100  -Addition in next Step

step: 2 Left Shift and Addition:
A: 1101  Q: 100_  -Unsuccessful
A: 1101  Q: 1000  -Addition in next Step

step: 3 Left Shift and Addition:
A: 1111  Q: 000_  -Unsuccessful
A: 1111  Q: 0000  -Addition in next Step

step: 4 Left Shift and Addition:
A: 0010  Q: 000_  Successful
A: 0010  Q: 0001  -Subtraction in next step

Quotient(Q): 0001  Remainder(A): 0010
PS C:\.vscode\college>
```

**CONCLUSION:** Thus, we have successfully implemented both the restoring division and the non-restoring division algorithm.