YASH SARANG
D6AD
47
DLCOA / Experiment 9

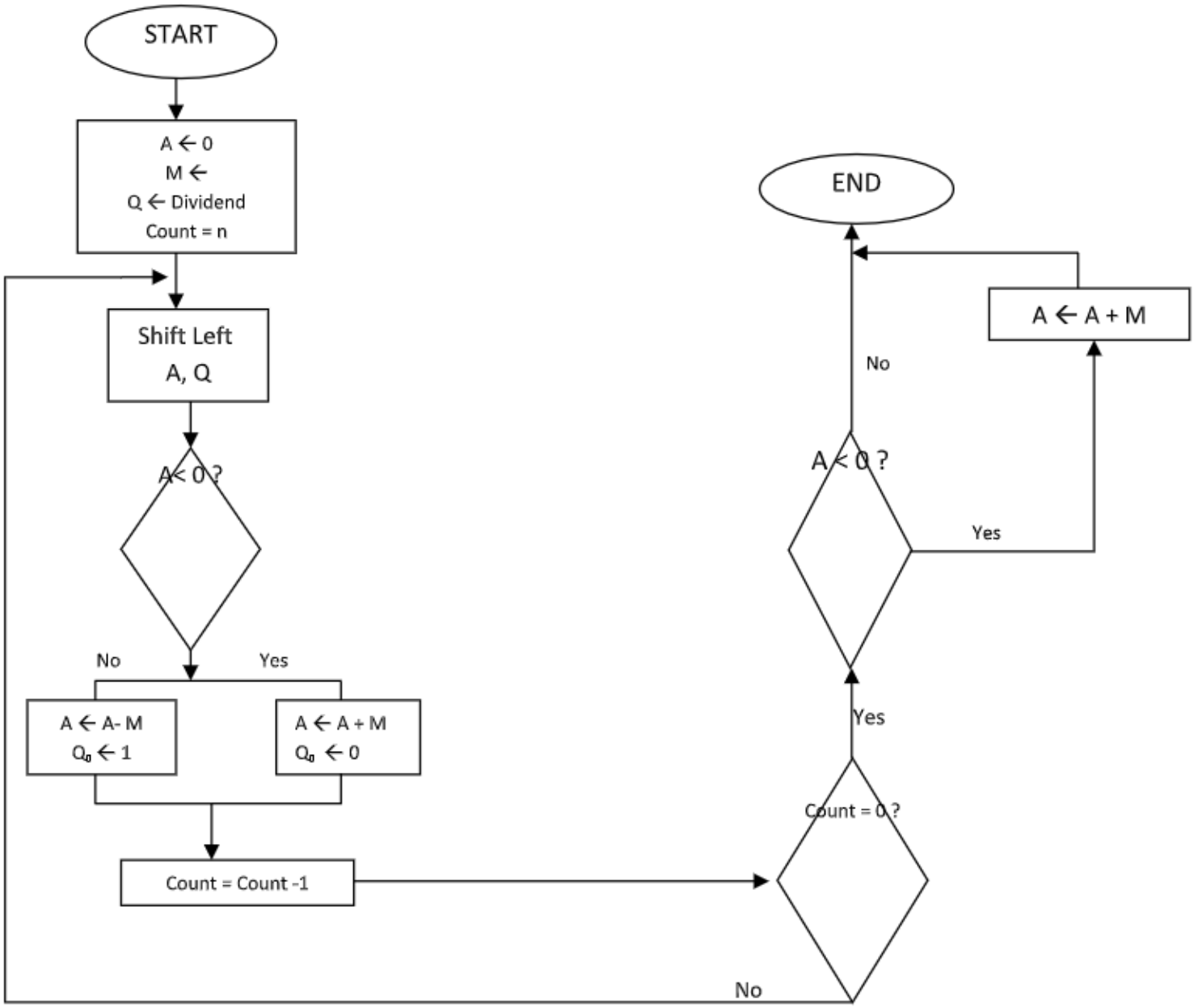
AIM: To write a C program for implementation of Non- Restoring Division.

SOFTWARE: Turbo C IDE

THEORY: In non-restoring division method, the dividend is NOT restored after the subtraction operation. Instead, the logic or flow of the code is as follows:

If the current remainder is positive then,
    Q. = 1;
Next operation will be shift and subtract;
Else if remainder is negative then,
    Q. = 0;
Next operation will be shift and add;

FLOWCHART:



CODE:

```
#include<stdio.h>
#include<stdlib.h>
```

```
int acum[100] = {0};
void add(int acum[], int b[], int n);
int q[100], b[100], l;
int main()
{
  int x, y;
  printf("Enter the Number : ");
  scanf("%d%d", & x, & y);
  int i = 0;

  while (x > 0 || y > 0) {
    if (x > 0)
    {
      q[i] = x % 2;
      x = x / 2;
    }
    else
    {
      q[i] = 0;
    }
    if (y > 0)
    {
      b[i] = y % 2;
      y = y / 2;
    }
    else
    {
      b[i] = 0;
    }
    i++;
  }

  int n = i;
  int bc[50];
  printf("\n");

  for (i = 0; i < n; i++)
  {
```

```
  if (b[i] == 0)
  {
    bc[i] = 1;
  }
  else
  {
    bc[i] = 0;
  }
}

bc[n] = 1;
for (i = 0; i <= n; i++)
{
  if (bc[i] == 0)
  {
    bc[i] = 1;
    i = n + 2;
  }
  else
  {
    bc[i] = 0;
  }
}

b[n] = 0;
int j;

for (i = n; i != 0; i--)
{
  if (acum[n] == 0)
  {
    for (j = n; j > 0; j--)
    {
      acum[j] = acum[j - 1];
    }
    acum[0] = q[n - 1];
    for (j = n - 1; j > 0; j--)
    {
      q[j] = q[j - 1];
```

```
    }
    add(acum, bc, n + 1);
  }
  else
  {
    for (j = n; j > 0; j--)
    {
      acum[j] = acum[j - 1];
    }
    acum[0] = q[n - 1];
    for (j = n - 1; j > 0; j--)
    {
      q[j] = q[j - 1];
    }
    add(acum, b, n + 1);
  }

  if (acum[n] == 1)
  {
    q[0] = 0;
  }
  else
  {
    q[0] = 1;
  }
}

if (acum[n] == 1)
{
  add(acum, b, n + 1);
}

printf("\nQuoient : ");
for (l = n - 1; l >= 0; l--)
{
  printf("%d", q[l]);
}

printf("\nRemainder : ");
```

```
  for (l = n; l >= 0; l--) {
    printf("%d", acum[l]);
  }
  return 0;
}
void add(int acum[], int bo[], int n) {
  int i = 0, temp = 0, sum = 0;
  for (i = 0; i < n; i++) {
    sum = 0;
    sum = acum[i] + bo[i] + temp;
    if (sum == 0) {
      acum[i] = 0;
      temp = 0;
    } else if (sum == 2) {
      acum[i] = 0;
      temp = 1;
    } else if (sum == 1) {
      acum[i] = 1;
      temp = 0;
    } else if (sum == 3) {
      acum[i] = 1;
      temp = 1;
    }
  }
}
```


OUTPUT:

```
Enter the Number  : 15
7


Quoient   : 0010
Remainder : 00001
```

