**Integer restoring Division**

**EXP NO: 33**

**AIM:** To write a C program to implement Integer restoring divison

.

**PROCEDURE:**

1. Initialize

- Set the dividend (numerator) as the initial remainder.

- Set the divisor (denominator) as the divisor.

2.Set Quotient and Remainder Registers:

- Initialize a quotient register to zero.

- Initialize a remainder register with the dividend.

3. Initialize the Counter:

- Set a counter to the number of bits in the divisor (or the dividend).

4. Shift:

- Shift the remainder and quotient registers left by one bit.

- Bring the next bit from the dividend into the least significant bit of the remainder register.

5. Subtract:

- Subtract the divisor from the shifted remainder.

- If the result is non-negative, set the MSB (Most Significant Bit) of the quotient to 1.

- If the result is negative, set the MSB of the quotient to 0 and add the divisor back to the remainder.

6. Check the Sign Bit:

- If the sign bit of the remainder is 0, the quotient is correct; otherwise, the quotient is incorrect.

7. Decrement Counter:

- Decrement the counter.

8. Repeat:

- If the counter is not zero, repeat steps 4-7.

9. Result:

- The content of the quotient register is the quotient.

- The content of the remainder register is the remainder.

**PROGRAM:**

#include <stdlib.h>

#include <stdio.h>

int acum[100] = {0};

int q[100], b[100];

void add(int acum[], int b[], int n);

int main() {

int x, y;

printf("Enter the Numbers: ");

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;

}

}

int l;

b[n] = 0;

int k = n;

int n1 = n + n - 1;

for (i = n; i != 0; i--) {

for (int j = n; j > 0; j--) {

acum[j] = acum[j - 1];

}

acum[0] = q[n - 1];

for (int j = n - 1; j > 0; j--) {

q[j] = q[j - 1];

}

add(acum, bc, n + 1);

if (acum[n] == 1) {

q[0] = 0;

add(acum, b, n + 1);

} else {

q[0] = 1;

}

}

printf("\nQuotient : ");

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 temp = 0, sum = 0;

for (int 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;

}

}

}

**INPUT:**

**A black and white text

Description automatically generated**

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

**RESULT:** Thus the program was executed successfully using DevC++.