**Name:Gaurav Kishor Patil**

**Roll No:54 Batch:C**

**Div:2**

|  |
| --- |
| Experiment No. 6 |
| Implement Booth’s algorithm using c-programming |
| Date of Performance:06/09/23 |
| Date of Submission:13/09/23 |

**Aim:** To implement Booth’s algorithm using c-programming.

# Objective -

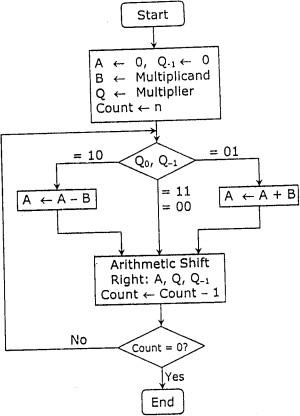
1. To understand the working of Booths algorithm.
2. To understand how to implement Booth’s algorithm using c-programming.

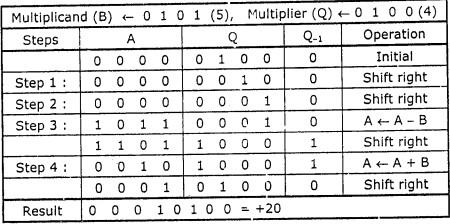
# Theory:

Booth’s algorithm is a multiplication algorithm that multiplies two signed binary numbers in 2’s complement notation. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed.

The algorithm works as per the following conditions :

1. If Qn and Q-1 are same i.e. 00 or 11 perform arithmetic shift by 1 bit.
2. If Qn Q-1 = 10 do A= A - B and perform arithmetic shift by 1 bit.
3. If Qn Q-1 = 01 do A= A + B and perform arithmetic shift by 1 bit.





# Program:

#include <stdio.h>

#include<conio.h>

#include <math.h>

int a = 0,b = 0, c = 0, a1 = 0, b1 = 0, com[5] = { 1, 0, 0, 0, 0};

int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};

int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};

void binary(){

int r,r2,i,temp;

a1 = fabs(a);

b1 = fabs(b);

for (i = 0; i < 5; i++){

r = a1 % 2;

a1 = a1 / 2;

r2 = b1 % 2;

b1 = b1 / 2;

anum[i] = r;

anumcp[i] = r;

bnum[i] = r2;

if(r2 == 0){

bcomp[i] = 1;

}

if(r == 0){

acomp[i] =1;

}

}

c = 0;

for ( i = 0; i < 5; i++){

res[i] = com[i]+ bcomp[i] + c;

if(res[i] >= 2){

c = 1;

}

else

c = 0;

res[i] = res[i] % 2;

}

for (i = 4; i >= 0; i--){

bcomp[i] = res[i];

}

//in case of negative inputs

if (a < 0){

c = 0;

for (i = 4; i >= 0; i--){

res[i] = 0;

}

for ( i = 0; i < 5; i++){

res[i] = com[i] + acomp[i] + c;

if (res[i] >= 2){

c = 1;

}

else

c = 0;

res[i] = res[i]%2;

}

for (i = 4; i >= 0; i--){

anum[i] = res[i];

anumcp[i] = res[i];

}

}

if(b < 0){

for (i = 0; i < 5; i++){

temp = bnum[i];

bnum[i] = bcomp[i];

bcomp[i] = temp;

}

}

}

void add(int num[]){

int i;

c = 0;

for ( i = 0; i < 5; i++){

res[i] = pro[i] + num[i] + c;

if (res[i] >= 2){

c = 1;

}

else{

c = 0;

}

res[i] = res[i]%2;

}

for (i = 4; i >= 0; i--){

pro[i] = res[i];

printf("%d",pro[i]);

}

printf(":");

for (i = 4; i >= 0; i--){

printf("%d", anumcp[i]);

}

}

void arshift(){//for arithmetic shift right

int temp = pro[4], temp2 = pro[0], i;

for (i = 1; i < 5 ; i++){

pro[i-1] = pro[i];

}

pro[4] = temp;

for (i = 1; i < 5 ; i++){

anumcp[i-1] = anumcp[i];

}

anumcp[4] = temp2;

printf("\nAR-SHIFT: ");

for (i = 4; i >= 0; i--){

printf("%d",pro[i]);

}

printf(":");

for(i = 4; i >= 0; i--){

printf("%d", anumcp[i]);

}

}

int main(){

int i, q = 0;

printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");

printf("\nEnter two numbers to multiply: ");

printf("\nBoth must be less than 16");

do{

printf("\nEnter A: ");

scanf("%d",&a);

printf("Enter B: ");

scanf("%d", &b);

}while(a >=16 || b >=16);

printf("\nExpected product = %d", a \* b);

binary();

printf("\n\nBinary Equivalents are: ");

printf("\nA = ");

for (i = 4; i >= 0; i--){

printf("%d", anum[i]);

}

printf("\nB = ");

for (i = 4; i >= 0; i--){

printf("%d", bnum[i]);

}

printf("\nB'+ 1 = ");

for (i = 4; i >= 0; i--){

printf("%d", bcomp[i]);

}

printf("\n\n");

for (i = 0;i < 5; i++){

if (anum[i] == q){

printf("\n-->");

arshift();

q = anum[i];

}

else if(anum[i] == 1 && q == 0){

printf("\n-->");

printf("\nSUB B: ");

add(bcomp);

arshift();

q = anum[i];

}

else{

printf("\n-->");

printf("\nADD B: ");

add(bnum);

arshift();

q = anum[i];

}

}

printf("\nProduct is = ");

for (i = 4; i >= 0; i--){

printf("%d", pro[i]);

}

for (i = 4; i >= 0; i--){

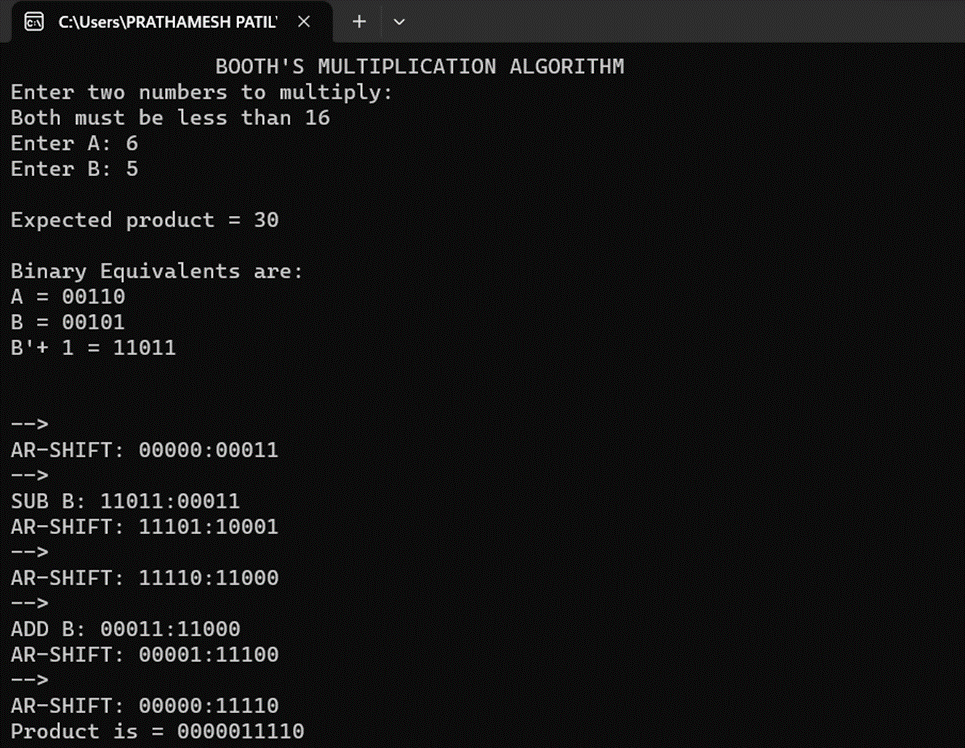
printf("%d", anumcp[i]);

}

return 0;

}

**Output:**

****

# Conclusion -

Booth’s algorithm is a multiplication algorithm that multiplies two signed binary numbers in two’s complement notation. It is used to perform multiplication of two signed numbers using binary arithmetic. The algorithm works by repeatedly adding or subtracting one of the two numbers based on the value of a third number called the multiplier. The multiplier is a binary number that is used to determine whether to add or subtract the second number. The algorithm uses a shift-and-add method to perform the multiplication. The C program to implement Booth’s algorithm for multiplication of two signed numbers using C programming language can be found online. The program takes two numbers as input and returns their product as output. The program output is also shown in the source code .