**BOOTH ALGORITHM**

**EXP NO: 34**

**AIM:** To write a C program to implement BOOTH ALGORITHM

**ALGORITHM:**

1. \*\*Initialize:\*\*

- Set two registers, `Multiplier` and `Multiplier Extension`, initially both holding the multiplier.

- Set another register, `Product`, initially with the multiplicand.

- Initialize a counter to the number of bits in the multiplier.

2. \*\*Loop:\*\*

- Repeat the following steps until the counter becomes zero.

3. \*\*Check LSB of Multiplier:\*\*

- Check the least significant bit (LSB) of the `Multiplier`.

4. \*\*If the Last Two Bits are 01:\*\*

- If the last two bits of `Multiplier` and `Multiplier Extension` are 01, add the `Product` to the result.

5. \*\*If the Last Two Bits are 10:\*\*

- If the last two bits of `Multiplier` and `Multiplier Extension` are 10, subtract the `Product` from the result.

6. \*\*Shift Right:\*\*

- Right shift `Multiplier` and `Multiplier Extension`, and also right shift the result.

7. \*\*Decrement Counter:\*\*

- Decrement the counter.

8. \*\*Repeat:\*\*

- If the counter is not zero, repeat steps 3-7.

9. \*\*Result:\*\*

- The content of the result register is the product of the multiplication.

Here's a simplified example to illustrate the Booth's Algorithm:

**PROGRAM:**

#include <stdio.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() {

a1 = fabs(a);

b1 = fabs(b);

int r, r2, i, temp;

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;

}

}

// Part for two's complementing

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() {

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");

// Simulating for two numbers each below 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) { // Just shift for 00 or 11

printf("\n-->");

arshift();

q = anum[i];

} else if (anum[i] == 1 && q == 0) { // Subtract and shift for 10

printf("\n-->");

printf("\nSUB B: ");

add(bcomp); // Add two's complement to implement subtraction

arshift();

q = anum[i];

} else { // Add and shift for 01

printf("\n-->");

printf("\nADD B: ");

add(bnum);

arshift();

q = anum[i];

}

}

printf("\nProduct is = ");

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

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

}

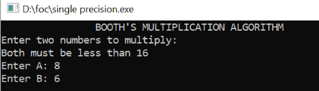
for (i = 4; i >= 0; i--) {

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

}

}

**INPUT:**

****

**OUTPUT:**

**A screenshot of a computer

Description automatically generated**

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