Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

| |
|---|
| Experiment No. 7 |
| 1mplement Booth's algorithm using c-programming |
| Name: Ankush Chavate |
| Roll Number: 5 |
| Date of Performance: |
| Date of Submission: |

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

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:

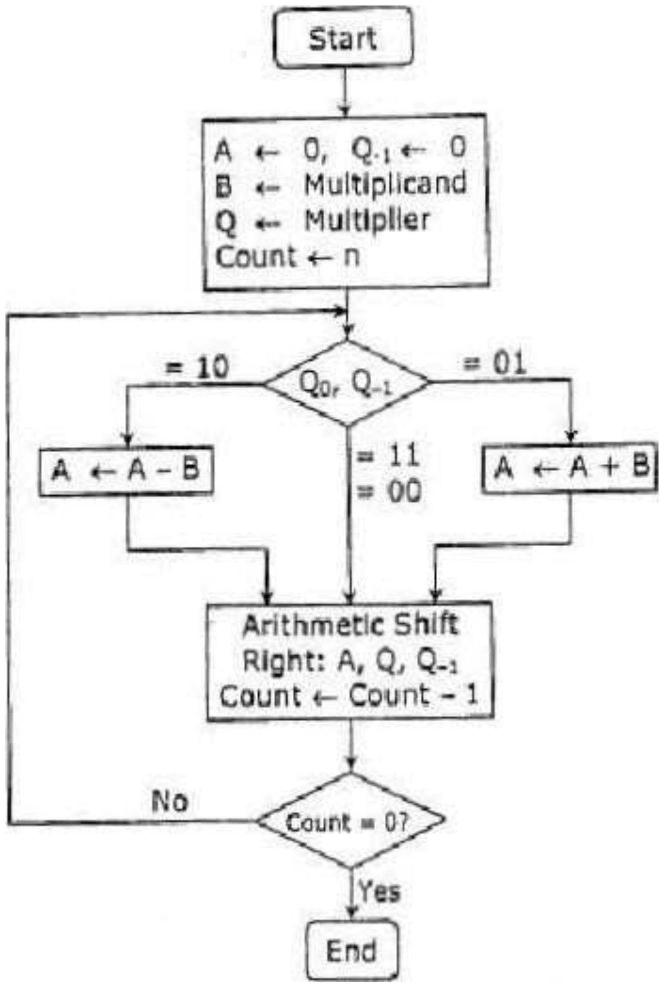
Booth's algorithm is a multiplication algorithm that multiplies two signed binary numbers in 2's complement notation. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. The algorithm works as per the following conditions :

l. IfQn and are same i.e. 00 or I I perform arithmetic shift by I bit.

2. IfQn Q = 10 do A= A - B and perform arithmetic shift by I bit.

3. IfQn 01 do A + B and perform arithmetic shift by I bit.

| Multiplicand (B) | | ɔ 1 | Multiplier (Q) | 1 0 0 (4) |
|---|---|---|---|---|
| Steps | | | | Operati o n |
| | o o o o | o 1 o o | | Initial |
| Step 1 | o o o o | o o 1 | | Shift right |
| Step 2 | o o o | o o o 1 | o | Shift right |
| step 3 | 1 1 | O o 1 | | |
| | 1 o 1 1 | 1 o o c | | Shift right |
| Step 4 : | o o 1 o | o o o | 1 | |
| | o o o | o 1 o o | | Shift right |
| Result | o o o | = +20 | | |




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;
     }
 }
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];
}
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++){//shift the MSB of product
       pro[i-1] = pro[i];
    }
    pro[4] = temp;
    for (i = 1; i < 5  ; i++){//shift the LSB of product
       anumcp[i-1] = anumcp[i];
    }
    anumcp[4] = temp2;
    printf("\nAR-SHIFT: ");//display together
```

```
    for (i = 4; i >= 0; i--){
        printf("%d",pro[i]);
    }
    printf(":");
    for(i = 4; i >= 0; i--){
        printf("%d", anumcp[i]);
    }
}

void 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){
         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{//add ans 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]);
    }
}
```

Output:

```
Both must be less than 16
Enter A: 2
Enter B: 4
Expected product = 8

Binary Equivalents are:
A = 00010
B = 00100
B'+ 1 = 11100


-->
AR-SHIFT: 00000:00001
-->
SUB B: 11100:00001
AR-SHIFT: 11110:00000
-->
ADD B: 00010:00000
AR-SHIFT: 00001:00000
-->
```

Conclusion - The aim of the experiment is to implement Booth's algorithm in C programming, a multiplication algorithm that efficiently and effectively multiplies two binary numbers using a sequential approach, reducing the number of partial products and improving computational speed.