|  |
| --- |
| Booth’s Multiplication Algorithm  Booth’s algorithm is a multiplication algorithm that multiplies two signed binary numbers in 2’s compliment notation. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth’s algorithm is of interest in the study of computer architecture. Here’s the implementation of the algorithm.  **Examples:**  Input : 0110, 0010  Output : qn q[n+1] AC QR sc(step count)  initial 0000 0010 4  0 0 rightShift 0000 0001 3  1 0 A = A - BR 1010  rightShift 1101 0000 2  0 1 A = A + BR 0011  rightShift 0001 1000 1  0 0 rightShift 0000 1100 0  Result=1100 // CPP code to implement booth's algorithm#include <bits/stdc++.h>using namespace std;// function to perform adding in the accumulatorvoid add(int ac[], int x[], int qrn){int i, c = 0;for (i = 0; i < qrn; i++) {// updating accumulator with A = A + BRac[i] = ac[i] + x[i] + c;if (ac[i] > 1) {ac[i] = ac[i] % 2;c = 1;}elsec = 0;}}// function to find the number's complementvoid complement(int a[], int n){int i;int x[8] = { 0};x[0] = 1;for (i = 0; i < n; i++) {a[i] = (a[i] + 1) % 2;}add(a, x, n);}// function ro perform right shiftvoid rightShift(int ac[], int qr[], int& qn, int qrn){int temp, i;temp = ac[0];qn = qr[0];cout << "\t\trightShift\t";for (i = 0; i < qrn - 1; i++) {ac[i] = ac[i + 1];qr[i] = qr[i + 1];}qr[qrn - 1] = temp;}// function to display oprationsvoid display(int ac[], int qr[], int qrn){int i;// accumulator contentfor (i = qrn - 1; i >= 0; i--)cout << ac[i];cout << "\t";// multiplier contentfor (i = qrn - 1; i >= 0; i--)cout << qr[i];}// Function to implement booth's algovoid boothAlgorithm(int br[], int qr[], int mt[], int qrn, int sc){int qn = 0, ac[10] = { 0 };int temp = 0;cout << "qn\tq[n+1]\t\tBR\t\tAC\tQR\t\tsc\n";cout << "\t\t\tinitial\t\t";display(ac, qr, qrn);cout << "\t\t" << sc << "\n";while (sc != 0) {cout << qr[0] << "\t" << qn;// SECOND CONDITIONif ((qn + qr[0]) == 1){if (temp == 0) {// subtract BR from accumulatoradd(ac, mt, qrn);cout << "\t\tA = A - BR\t";for (int i = qrn - 1; i >= 0; i--)cout << ac[i];temp = 1;}// THIRD CONDITIONelse if (temp == 1){// add BR to accumulatoradd(ac, br, qrn);cout << "\t\tA = A + BR\t";for (int i = qrn - 1; i >= 0; i--)cout << ac[i];temp = 0;}cout << "\n\t";rightShift(ac, qr, qn, qrn);}// FIRST CONDITIONelse if (qn - qr[0] == 0)rightShift(ac, qr, qn, qrn);display(ac, qr, qrn);cout << "\t";// decrement countersc--;cout << "\t" << sc << "\n";}}// driver codeint main(int argc, char\*\* arg){int mt[10], sc;int brn, qrn;// Number of multiplicand bitbrn = 4;// multiplicandint br[] = { 0, 1, 1, 0 };// copy multiplier to temp array mt[]for (int i = brn - 1; i >= 0; i--)mt[i] = br[i];reverse(br, br + brn);complement(mt, brn);// No. of multiplier bitqrn = 4;// sequence countersc = qrn;// multiplierint qr[] = { 1, 0, 1, 0 };reverse(qr, qr + qrn);boothAlgorithm(br, qr, mt, qrn, sc);cout << endl<< "Result = ";for (int i = qrn - 1; i >= 0; i--)cout << qr[i];} **Output :**  qn q[n + 1] BR AC QR sc  initial 0000 1010 4  0 0 rightShift 0000 0101 3  1 0 A = A - BR 1010  rightShift 1101 0010 2  0 1 A = A + BR 0011  rightShift 0001 1001 1  1 0 A = A - BR 1011  rightShift 1101 1100 0  Result = 1100 |

|  |
| --- |
|  |