1. **To find 1’s and 2’s Complement using C or C++.**

#include <stdio.h>

int main()

{

int n; // variable declaration

printf("Enter the number of bits do you want to enter :");

scanf("%d",&n);

char binary[n+1]; // binary array declaration;

char onescomplement[n+1]; // onescomplement array declaration

char twoscomplement[n+1]; // twoscomplement array declaration

int carry=1; // variable initialization

printf("\nEnter the binary number : ");

scanf("%s", binary);

printf("\nThe ones complement of the binary number is :");

// Finding onescomplement in C

for(int i=0;i<n;i++)

{

if(binary[i]=='0')

onescomplement[i]='1';

else if(binary[i]=='1')

onescomplement[i]='0';

}

onescomplement[n]='\0';

printf("%s",onescomplement);

printf("\nThe twos complement of the binary number is : ");

// Finding twoscomplement in C

for(int i=n-1; i>=0; i--)

{

if(onescomplement[i] == '1' && carry == 1)

{

twoscomplement[i] = '0';

}

else if(onescomplement[i] == '0' && carry == 1)

{

twoscomplement[i] = '1';

carry = 0;

}else

{

twoscomplement[i] = onescomplement[i];

}

}

twoscomplement[n]='\0';

printf("%s",twoscomplement);

return 0;

}

**2: Signed Magnitude multiplication**

#include<stdio.h>

#include<conio.h>

#include<math.h>

int q=0,b=0,c=0,e=0,q1=0,b1=0;

int qnum[4] = {0},bnum[4] = {0};

int acc[4] = {0}, res[4] = {0};

int sc=0,bs=0,qs=0,asize=0;

void binary()

{

b1 = fabs(b);

q1 = fabs(q);

int r1, r2, i, temp;

for (i = 0; i < 4; i++)

{

r1 = b1 % 2;

b1 = b1 / 2;

r2 = q1 % 2;

q1 = q1 / 2;

bnum[i] = r1;

qnum[i] = r2;

}

sc=sizeof(qnum)/sizeof(int);

asize=sc;

}

void add(int num[])

{

int i;

c = 0;

for (i = 0; i < 4; i++){

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

if (res[i] >= 2){

c = 1;

}

else{

c = 0;

}

e=c;

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

}

for (i = 3; i >= 0; i--){

acc[i] = res[i];

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

}

printf(" : ");

for (i = 3; i >= 0; i--)

{

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

}

}

void rshift(){//for shift right

int temp2 = acc[0], i;

for (i = 1; i < 4 ; i++){//shift the MSB of product

acc[i-1] = acc[i];

}

acc[3] = e;

e=0;

for (i = 1; i < 4 ; i++){//shift the LSB of product

qnum[i-1] = qnum[i];

}

qnum[3] = temp2;

printf("\n R-SHIFT: ");//display together

for (i = 3; i >= 0; i--){

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

}

printf(":");

for(i = 3; i >= 0; i--){

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

}

}

int main()

{

int i;

int p=0,n=1;

printf("\t\tSIGNED MAGNITUDE 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 b: ");

scanf("%d",&b);

printf("Enter Q: ");

scanf("%d",&q);

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

printf("\n Expected product = %d", b \* q);

binary();

printf("\nS.C. = %d",sc);

printf("\n\n Signed Binary Equivalents are: ");

printf("\n b = ");

for (i = 3; i >= 0; i--){

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

}

printf("\n q = ");

for (i = 3; i >= 0; i--){

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

}

printf("\n\n");

while(sc!=0)

{

printf("\nS.C. = %d",sc);

if(qnum[0]==0){

printf("\n-->");

rshift();

}

else

{

printf("\n-->");

printf("\n ADD B: ");

add(bnum);

rshift();

}

sc--;

}

printf("\nproduct is = ");

if((b<0 && q>0)||(b>0 && q<0))

printf("%d",n);

else

printf("%d",p);

for (i = 3; i >= 0; i--){

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

}

for (i = 3; i >= 0; i--){

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

}

return 0;

}

**Program for Implementing Multiplication**

#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(){//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){//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 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]);

}

}

**/\*****Program for implementing Restoring Division algorithm.\*/**

#include <stdio.h>

#include <math.h>

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

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

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

void binary(){

a = fabs(a);

b = fabs(b);

int r, r2, i, temp;

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

r = a % 2;

a = a / 2;

r2 = b % 2;

b = b / 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];

}

}

void add(int num[]){

int i;

c = 0;

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

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

if(res[i]>=2){

c = 1;

}

else

c = 0;

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

}

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

rem[i] = res[i];

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

}

printf(":");

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

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

}

}

void shl(){//for shift left

int i;

for(i = 4; i > 0 ; i--){//shift the remainder

rem[i] = rem[i-1];

}

rem[0] = anumcp[4];

for(i = 4; i > 0 ; i--){//shift the remtient

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

}

anumcp[0] = 0;

printf("\nSHIFT LEFT: ");//display together

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

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

}

printf(":");

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

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

}

}

int main(){

int i;

printf("\t\tRESTORING DIVISION ALGORITHM");

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

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 Quotient = %d", a/b);

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

if(a\*b <0){

s = 1;

}

binary();

printf("\n\nUnsigned Binary 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-->");

//division part

shl();

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

printf("\n-->"); //start with subtraction

printf("\nSUB B: ");

add(bcomp);

if(rem[4]==1){//simply add for restoring

printf("\n-->RESTORE");

printf("\nADD B: ");

anumcp[0] = 0;

add(bnum);

}

else{

anumcp[0] = 1;

}

if(i<4)

shl();

}

printf("\n----------------------------");

printf("\nSign of the result = %d",s);

printf("\nRemainder is = ");

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

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

}

printf("\nQuotient is = ");

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

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

}

}

**C program for implementing Booths Algorithm**

#include <iostream>

#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]);

}

// for arithmetic shift right

void arshift()

{

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

// shift the MSB of product

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

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]);

}

}

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 ans shift for 01

printf("\n-->");

printf("\nADD B: ");

add(bnum);

arshift();

q = anum[i];

}

}

printf("\nProduct: ");

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

{

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

}

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

{

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

}

}

1. Write the program for following statement by using three, single, zero address instructions.

X = (A+B\*C-D)/(E\*F+G)