**CODE:**

import java.util.\*;

class Multiplication

{

int n=0,carry=0;

int a[]=new int[8];

int b[]=new int[8];

int q[]=new int[8];

public static void main(String args[])

{

int i,x,y;

Multiplication m = new Multiplication();

Scanner scr=new Scanner(System.in);

System.out.print("Enter the number1: ");

x=scr.nextInt();

System.out.print("Enter the number2: ");

y=scr.nextInt();

for(i=0;i<m.a.length;i++)

{

m.a[i]=0;

}

m.toBinary(x,y);

if((m.n)<=4)

{

System.out.println("C\t A\t Q\t B");

System.out.println("--------------------------------");

}

else

{

System.out.println("C\t A\t\t Q\t\t B");

System.out.println("--------------------------------------------------");

}

System.out.print(m.carry+"\t");

for(i=m.a.length-m.n;i<m.a.length;i++)

{

System.out.print(m.a[i]);

}

System.out.print("\t");

for(i=m.q.length-m.n;i<m.q.length;i++)

{

System.out.print(m.q[i]);

}

System.out.print("\t");

for(i=m.b.length-m.n;i<m.b.length;i++)

{

System.out.print(m.b[i]);

}

System.out.println();

if(m.n<=4)

{

System.out.println("--------------------------------");

}

else

{

System.out.println("--------------------------------------------------");

}

m.multiply();

m.toDecimal();

}

public void toBinary(int x, int y)

{

int rem,index=7,n1=0,n2=0;

while(x>0)

{

rem=x%2;

b[index--]=rem;

x=x/2;

n1++;

}

index=7;

while(y>0)

{

rem=y%2;

q[index--]=rem;

y=y/2;

n2++;

}

if(n1>n2)

n=n1;

else

n=n2;

}

public void multiply()

{

int i,j;

for(j=n;j>0;j--)

{

if(q[7]==1)

add();

System.out.print(carry+"\t");

for(i=a.length-n;i<a.length;i++)

{

System.out.print(a[i]);

}

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

{

System.out.print(q[i]);

}

System.out.print("\t");

for(i=b.length-n;i<b.length;i++)

{

System.out.print(b[i]);

}

System.out.println();

rshift();

}

}

public void add()

{

int i;

for(i=a.length-1;i>7-n;i--)

{

carry=0;

a[i]=a[i]+b[i];

if(a[i]==2)

{

a[i]=0;

carry=1;

}

else if(a[i]==3)

{

a[i]=1;

carry=1;

}

if(i>n)

a[i-1]=a[i-1]+carry;

}

}

public void rshift()

{

int i;

for(i=7;i>8-n;i--)

q[i]=q[i-1];

q[8-n]=a[7];

for(i=7;i>0;i--)

a[i]=a[i-1];

a[8-n]=carry;

carry=0;

System.out.print(carry+"\t");

for(i=a.length-n;i<a.length;i++)

{

System.out.print(a[i]);

}

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

{

System.out.print(q[i]);

}

System.out.print("\t");

for(i=b.length-n;i<q.length;i++)

{

System.out.print(b[i]);

}

System.out.println();

if(n<=4)

{

System.out.println("--------------------------------");

}

else

{

System.out.println("--------------------------------------------------");

}

}

public void toDecimal()

{

int ans=0,i,count=0;;

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

{

ans=ans+(int)(q[7-i]\*Math.pow(2,count++));

}

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

{

ans=ans+(int)(a[7-i]\*Math.pow(2,count++));

}

System.out.println("Answer is: "+ans);

}

}

**OUTPUT:**

Enter the number1: 9

Enter the number2: 12

C A Q B

--------------------------------

0 0000 1100 1001

--------------------------------

0 0000 1100 1001

0 0000 0110 1001

--------------------------------

0 0000 0110 1001

0 0000 0011 1001

--------------------------------

0 1001 0011 1001

0 0100 1001 1001

--------------------------------

0 1101 1001 1001

0 0110 1100 1001

--------------------------------

Answer is: 108

**CODE:**

import java.util.\*;

class Booths

{

int n=0,q\_1=0,carry=0;

int a[]=new int[8];

int b[]=new int[8];

int q[]=new int[8];

public static void main(String args[])

{

int i,x,y;

Booths booths = new Booths();

Scanner scr=new Scanner(System.in);

System.out.print("Enter the number1: ");

x=scr.nextInt();

System.out.print("Enter the number2: ");

y=scr.nextInt();

for(i=0;i<booths.a.length;i++)

{

booths.a[i]=0;

}

booths.toBinary(x,y);

if((booths.n)<=4)

{

System.out.println("A\t Q\tQ-1\t B");

System.out.println("--------------------------------");

}

else

{

System.out.println("A\t Q\t\t Q-1\t\t B");

System.out.println("----------------------------------------------");

}

for(i=booths.a.length-booths.n;i<booths.a.length;i++)

{

System.out.print(booths.a[i]);

}

System.out.print("\t");

for(i=booths.q.length-booths.n;i<booths.q.length;i++)

{

System.out.print(booths.q[i]);

}

System.out.print("\t");

System.out.print(+booths.q\_1+"\t");

for(i=booths.b.length-booths.n;i<booths.b.length;i++)

{

System.out.print(booths.b[i]);

}

System.out.println();

if(booths.n<=4)

{

System.out.println("--------------------------------");

}

else

{

System.out.println("--------------------------------------------------");

}

booths.multiply();

booths.toDecimal();

}

public void toBinary(int x, int y)

{

int rem,index=7,n1=0,n2=0,x1,y1,i;

x1=x;

y1=y;

x=Math.abs(x);

y=Math.abs(y);

while(x>0)

{

rem=x%2;

b[index--]=rem;

x=x/2;

n1++;

}

index=7;

while(y>0 || y<0)

{

rem=y%2;

q[index--]=rem;

y=y/2;

n2++;

}

if(n1<=4 || n2<=4)

n=4;

else

n=8;

if(x1<0)

{

twos\_complement(b);

}

if(y1<0)

{

twos\_complement(q);

}

}

public void ones\_complement(int a[])

{

int i;

for(i=a.length-n;i<a.length;i++)

{

if(a[i]==0)

a[i]=1;

else if(a[i]==1)

a[i]=0;

}

}

public void twos\_complement(int a[])

{

int i;

int c[]=new int[8];

c[7]=1;

for(i=a.length-n;i<a.length;i++)

{

if(a[i]==0)

a[i]=1;

else if(a[i]==1)

a[i]=0;

}

add(a,c);

}

public void multiply()

{

int i,j;

int copy[]=new int[8];

for(j=n;j>0;j--)

{

if((q[7]==0 && q\_1==0) || (q[7]==1 && q\_1==1))

rshift();

else if((q[7]==0 && q\_1==1))

{

add(a,b);

for(i=a.length-n;i<a.length;i++)

{

System.out.print(a[i]);

}

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

{

System.out.print(q[i]);

}

System.out.print("\t");

System.out.print(+q\_1+"\t");

for(i=b.length-n;i<b.length;i++)

{

System.out.print(b[i]);

}

System.out.println();

rshift();

}

else if((q[7]==1 && q\_1==0))

{

for(i=0;i<b.length;i++)

copy[i]=b[i];

twos\_complement(copy);

add(a,copy);

for(i=a.length-n;i<a.length;i++)

{

System.out.print(a[i]);

}

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

{

System.out.print(q[i]);

}

System.out.print("\t");

System.out.print(+q\_1+"\t");

for(i=b.length-n;i<b.length;i++)

{

System.out.print(b[i]);

}

System.out.println();

rshift();

}

}

}

public void add(int a[],int b[])

{

int i;

for(i=a.length-1;i>7-n;i--)

{

carry=0;

a[i]=a[i]+b[i];

if(a[i]==2)

{

a[i]=0;

carry=1;

}

else if(a[i]==3)

{

a[i]=1;

carry=1;

}

if(i>n)

a[i-1]=a[i-1]+carry;

}

}

public void rshift()

{

int i;

q\_1=q[7];

for(i=7;i>8-n;i--)

q[i]=q[i-1];

q[8-n]=a[7];

for(i=7;i>8-n;i--)

a[i]=a[i-1];

for(i=a.length-n;i<a.length;i++)

{

System.out.print(a[i]);

}

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

{

System.out.print(q[i]);

}

System.out.print("\t");

System.out.print(+q\_1+"\t");

for(i=b.length-n;i<b.length;i++)

{

System.out.print(b[i]);

}

System.out.println();

if(n<=4)

{

System.out.println("--------------------------------");

}

else

{

System.out.println("--------------------------------------------------");

}

}

public void toDecimal()

{

int ans=0,i,count=0;

boolean flag=false;

if(a[8-n]==1)

{

ones\_complement(a);

twos\_complement(q);

flag=true;

}

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

{

ans=ans+(int)(q[7-i]\*Math.pow(2,count++));

}

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

{

ans=ans+(int)(a[7-i]\*Math.pow(2,count++));

}

if(flag)

ans=-ans;

System.out.println("Answer is: "+ans);

}

}

**OUTPUT:**

Enter the number1: -7

Enter the number2: 6

A Q Q-1 B

--------------------------------

0000 0110 0 1001

--------------------------------

0000 0011 0 1001

--------------------------------

0111 0011 0 1001

0011 1001 1 1001

--------------------------------

0001 1100 1 1001

--------------------------------

1010 1100 1 1001

1101 0110 0 1001

--------------------------------

Answer is: -42

**CODE:**

import java.util.\*;

class RestoringDivision

{

int n=0,q\_1=0,carry=0;

int a[]=new int[8];

int b[]=new int[8];

int q[]=new int[8];

public static void main(String args[])

{

int i,x,y;

RestoringDivision d= new RestoringDivision();

Scanner scr=new Scanner(System.in);

System.out.print("Enter the dividend: ");

y=scr.nextInt();

System.out.print("Enter the divisor: ");

x=scr.nextInt();

for(i=0;i<d.a.length;i++)

d.a[i]=0;

d.toBinary(x,y);

if((d.n)<=4)

{

System.out.println(" A\t Q\t M\t COMMENT");

System.out.println("-----------------------------------------------");

}

else

{

System.out.println(" A\t\t Q\t\t M\t\t COMMENT");

System.out.println("-----------------------------------------------------------------------");

}

d.printArray("Initialize");

if((d.n)<=4)

System.out.println("-----------------------------------------------");

else

System.out.println("-----------------------------------------------------------------------");

d.divide();

d.toDecimal();

}

public void printArray(String s)

{

int i;

for(i=a.length-n;i<a.length;i++)

System.out.print(a[i]);

System.out.print("\t");

for(i=q.length-n;i<q.length;i++)

System.out.print(q[i]);

System.out.print("\t");

for(i=b.length-n;i<b.length;i++)

System.out.print(b[i]);

System.out.println("\t"+s);

}

public void toBinary(int x, int y)

{

int rem,index=7,n1=0,n2=0,x1,y1,i;

x1=x;

y1=y;

x=Math.abs(x);

y=Math.abs(y);

while(x>0)

{

rem=x%2;

b[index--]=rem;

x=x/2;

n1++;

}

index=7;

while(y>0 || y<0)

{

rem=y%2;

q[index--]=rem;

y=y/2;

n2++;

}

if(n1<=4 && n2<=4)

n=4;

else

n=8;

if(x1<0)

twos\_complement(b);

if(y1<0)

twos\_complement(q);

}

public void ones\_complement(int a[])

{

int i;

for(i=a.length-n;i<a.length;i++)

{

if(a[i]==0)

a[i]=1;

else if(a[i]==1)

a[i]=0;

}

}

public void twos\_complement(int a[])

{

int c[]=new int[8];

c[7]=1;

ones\_complement(a);

add(a,c);

}

public void divide()

{

int i,j;

int copy[]=new int[8];

for(j=n;j>0;j--)

{

lshift();

for(i=0;i<b.length;i++)

copy[i]=b[i];

twos\_complement(copy);

add(a,copy);

printArray("Subtract");

if(a[8-n]==1)

{

q[7]=0;

add(a,b);

printArray("Restore and set Q0=0");

}

else

{

q[7]=1;

printArray("Q0=1");

}

if((n)<=4)

System.out.println("-----------------------------------------------");

else

System.out.println("-----------------------------------------------------------------------");

}

}

public void add(int a[],int b[])

{

int i;

for(i=a.length-1;i>=(8-n);i--)

{

carry=0;

a[i]=a[i]+b[i];

if(a[i]==2)

{

a[i]=0;

carry=1;

}

else if(a[i]==3)

{

a[i]=1;

carry=1;

}

if(i>0)

a[i-1]=a[i-1]+carry;

}

}

public void lshift()

{

int i;

for(i=8-n;i<7;i++)

a[i]=a[i+1];

a[7]=q[8-n];

for(i=8-n;i<7;i++)

q[i]=q[i+1];

q[7]=0;

printArray("Left Shift");

}

public void toDecimal()

{

int quotient=0,i,count=0,remainder=0;

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

quotient=quotient+(int)(q[7-i]\*Math.pow(2,count++));

count=0;

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

remainder=remainder+(int)(a[7-i]\*Math.pow(2,count++));

System.out.println("Quotient is: "+quotient);

System.out.println("Remainder is: "+remainder);

}

}

**OUTPUT:**

Enter the dividend: 35

Enter the divisor: 4

A Q M COMMENT

-----------------------------------------------------------------------

00000000 00100011 00000100 Initialize

-----------------------------------------------------------------------

00000000 01000110 00000100 Left Shift

11111100 01000110 00000100 Subtract

00000000 01000110 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000000 10001100 00000100 Left Shift

11111100 10001100 00000100 Subtract

00000000 10001100 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000001 00011000 00000100 Left Shift

11111101 00011000 00000100 Subtract

00000001 00011000 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000010 00110000 00000100 Left Shift

11111110 00110000 00000100 Subtract

00000010 00110000 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000100 01100000 00000100 Left Shift

00000000 01100000 00000100 Subtract

00000000 01100001 00000100 Q0=1

-----------------------------------------------------------------------

00000000 11000010 00000100 Left Shift

11111100 11000010 00000100 Subtract

00000000 11000010 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000001 10000100 00000100 Left Shift

11111101 10000100 00000100 Subtract

00000001 10000100 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

00000011 00001000 00000100 Left Shift

11111111 00001000 00000100 Subtract

00000011 00001000 00000100 Restore and set Q0=0

-----------------------------------------------------------------------

Quotient is: 8

Remainder is: 3

**CODE:**

import java.util.\*;

class NRestoringDivision

{

int n=0,q\_1=0,carry=0;

int a[]=new int[8];

int b[]=new int[8];

int q[]=new int[8];

public static void main(String args[])

{

int i,x,y;

NRestoringDivision d= new NRestoringDivision();

Scanner scr=new Scanner(System.in);

System.out.print("Enter the dividend: ");

y=scr.nextInt();

System.out.print("Enter the divisor: ");

x=scr.nextInt();

for(i=0;i<d.a.length;i++)

d.a[i]=0;

d.toBinary(x,y);

if((d.n)<=5)

{

System.out.println("C\t A\t Q\t M\t COMMENT");

System.out.println("-----------------------------------------------");

}

else

{

System.out.println("C\tA\t\t Q\t\t M\t\t COMMENT");

System.out.println("-----------------------------------------------------------------------");

}

d.printArray("Initialize");

if((d.n)<=5)

System.out.println("-----------------------------------------------");

else

System.out.println("-----------------------------------------------------------------------");

d.divide();

d.toDecimal();

}

public void printArray(String s)

{

int i;

System.out.print(a[8-n]+"\t");

for(i=a.length-n+1;i<a.length;i++)

System.out.print(a[i]);

System.out.print("\t");

for(i=q.length-n+1;i<q.length;i++)

System.out.print(q[i]);

System.out.print("\t");

for(i=b.length-n+1;i<b.length;i++)

System.out.print(b[i]);

System.out.println("\t"+s);

}

public void toBinary(int x, int y)

{

int rem,index=7,n1=0,n2=0,x1,y1,i;

x1=x;

y1=y;

x=Math.abs(x);

y=Math.abs(y);

while(x>0)

{

rem=x%2;

b[index--]=rem;

x=x/2;

n1++;

}

index=7;

while(y>0 || y<0)

{

rem=y%2;

q[index--]=rem;

y=y/2;

n2++;

}

if(n1<=4 && n2<=4)

n=5;

else

n=8;

if(x1<0)

twos\_complement(b);

if(y1<0)

twos\_complement(q);

}

public void ones\_complement(int a[])

{

int i;

for(i=a.length-n;i<a.length;i++)

{

if(a[i]==0)

a[i]=1;

else if(a[i]==1)

a[i]=0;

}

}

public void twos\_complement(int a[])

{

int i;

int c[]=new int[8];

c[7]=1;

ones\_complement(a);

add(a,c);

}

public void divide()

{

int i,j;

int copy[]=new int[8];

for(j=n-1;j>0;j--)

{

if(a[8-n]==1)

{

lshift();

add(a,b);

printArray("A=A+M");

}

else

{

lshift();

for(i=0;i<b.length;i++)

copy[i]=b[i];

twos\_complement(copy);

add(a,copy);

printArray("A=A-M");

}

if(a[8-n]==1)

{

q[7]=0;

printArray("Q0=0");

}

else

{

q[7]=1;

printArray("Q0=1");

}

if((n)<=5)

System.out.println("-----------------------------------------------");

else

System.out.println("-----------------------------------------------------------------------");

}

if(a[8-n]==1)

{

add(a,b);

printArray("Last A=A+M");

}

}

public void add(int a[],int b[])

{

int i;

for(i=a.length-1;i>=(8-n);i--)

{

carry=0;

a[i]=a[i]+b[i];

if(a[i]==2)

{

a[i]=0;

carry=1;

}

else if(a[i]==3)

{

a[i]=1;

carry=1;

}

if(i>0)

a[i-1]=a[i-1]+carry;

}

}

public void lshift()

{

int i,copy;

for(i=8-n;i<7;i++)

a[i]=a[i+1];

a[7]=q[8-n+1];

for(i=8-n+1;i<7;i++)

q[i]=q[i+1];

q[7]=0;

printArray("Left Shift");

}

public void toDecimal()

{

int quotient=0,i,count=0,remainder=0;

boolean flaga=false,flagq=false;

if(a[8-n]==1)

{

twos\_complement(a);

flaga=true;

}

if(q[8-n+1]==1)

{

twos\_complement(q);

flagq=true;

}

for(i=0;i<n-1;i++)

{

quotient=quotient+(int)(q[7-i]\*Math.pow(2,count++));

}

count=0;

for(i=0;i<n-1;i++)

{

remainder=remainder+(int)(a[7-i]\*Math.pow(2,count++));

}

if(flaga)

quotient=-quotient;

if(flagq)

remainder=-remainder;

System.out.println("Quotient is: "+quotient);

System.out.println("Remainder is: "+remainder);

}

}

**OUTPUT:**

Enter the dividend: 10

Enter the divisor: 3

C A Q M COMMENT

-----------------------------------------------

0 0000 1010 0011 Initialize

-----------------------------------------------

0 0001 0100 0011 Left Shift

1 1110 0100 0011 A=A-M

1 1110 0100 0011 Q0=0

-----------------------------------------------

1 1100 1000 0011 Left Shift

1 1111 1000 0011 A=A+M

1 1111 1000 0011 Q0=0

-----------------------------------------------

1 1111 0000 0011 Left Shift

0 0010 0000 0011 A=A+M

0 0010 0001 0011 Q0=1

-----------------------------------------------

0 0100 0010 0011 Left Shift

0 0001 0010 0011 A=A-M

0 0001 0011 0011 Q0=1

-----------------------------------------------

Quotient is: 3

Remainder is: 1

**CODE:**

cache\_size = input("Enter Cache Size: ")

cache = []

queue = []

print("Element\t"),

print("Description"),

print("Cache\t")

while(True):

x = input()

if( x in cache ):

queue.remove(x)

queue.append(x)

print("\tCache Hit\t"),

elif( x==-1 ):

break

else:

print("\tPage Fault\t"),

if( len(cache) == cache\_size ):

temp = queue.pop(0)

queue.append(x)

loc = cache.index(temp)

cache.insert( loc,x )

cache.remove(temp)

else:

cache.append(x)

queue.append(x)

print( cache )

**OUTPUT:**

Enter Cache Size: 3

Element Description Cache

7

       Page Fault      [7]

0

       Page Fault      [7, 0]

1

       Page Fault      [7, 0, 1]

2

       Page Fault      [2, 0, 1]

0

       Cache Hit       [2, 0, 1]

3

       Page Fault      [2, 0, 3]

**CODE:**

cache\_size = input("Enter Cache Size: ")

cache = []

print("Element\t"),

print("Description"),

print("Cache\t")

while(True):

x = input()

if( x in cache ):

print("\tCache Hit\t"),

elif( x==-1 ):

break

else:

print("\tPage Fault\t"),

if( len(cache) == cache\_size ):

cache.pop(0)

cache.insert( 0,x )

else:

cache.append(x)

print( cache )

**OUTPUT:**

Enter Cache Size: 3

Element Description Cache

7

       Page Fault      [7]

0

       Page Fault      [7, 0]

1

       Page Fault      [7, 0, 1]

7

       Cache Hit       [7, 0, 1]

2

       Page Fault      [2, 0, 1]

0

       Cache Hit       [2, 0, 1]

**CODE:**

import java.util.\*;

class Hamming {

public static void main(String args[]) {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the number of bits for the Hamming data:");

int n = scan.nextInt();

int a[] = new int[n];

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

System.out.println("Enter bit no. " + (n-i) + ":");

a[n-i-1] = scan.nextInt();

}

System.out.println("You entered:");

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

System.out.print(a[n-i-1]);

}

System.out.println();

int b[] = generateCode(a);

System.out.println("Generated code is:");

for(int i=0 ; i < b.length ; i++) {

System.out.print(b[b.length-i-1]);

}

System.out.println();

System.out.println("Enter position of a bit to alter to check for error detection at the receiver end (0 for no error):");

int error = scan.nextInt();

if(error != 0) {

b[error-1] = (b[error-1]+1)%2;

}

System.out.println("Sent code is:");

for(int i=0 ; i < b.length ; i++) {

System.out.print(b[b.length-i-1]);

}

System.out.println();

receive(b, b.length - a.length);

}

static int[] generateCode(int a[]) {

int b[];

int i=0, parity\_count=0 ,j=0, k=0;

while(i < a.length) {

if(Math.pow(2,parity\_count) == i+parity\_count + 1) {

parity\_count++;

}

else {

i++;

}

}

b = new int[a.length + parity\_count];

for(i=1 ; i <= b.length ; i++) {

if(Math.pow(2, j) == i) {

b[i-1] = 2;

j++;

}

else {

b[k+j] = a[k++];

}

}

for(i=0 ; i < parity\_count ; i++) {

b[((int) Math.pow(2, i))-1] = getParity(b, i);

}

return b;

}

static int getParity(int b[], int power) {

int parity = 0;

for(int i=0 ; i < b.length ; i++) {

if(b[i] != 2) {

int k = i+1;

String s = Integer.toBinaryString(k);

int x = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;

if(x == 1) {

if(b[i] == 1) {

parity = (parity+1)%2;

}

}

}

}

return parity;

}

static void receive(int a[], int parity\_count) {

int power;

int parity[] = new int[parity\_count];

String syndrome = new String();

for(power=0 ; power < parity\_count ; power++) {

for(int i=0 ; i < a.length ; i++) {

int k = i+1;

String s = Integer.toBinaryString(k);

int bit = ((Integer.parseInt(s))/((int) Math.pow(10, power)))%10;

if(bit == 1) {

if(a[i] == 1) {

parity[power] = (parity[power]+1)%2;

}

}

}

syndrome = parity[power] + syndrome;

}

int error\_location = Integer.parseInt(syndrome, 2);

if(error\_location != 0) {

System.out.println("Error is at location " + error\_location + ".");

a[error\_location-1] = (a[error\_location-1]+1)%2;

System.out.println("Corrected code is:");

for(int i=0 ; i < a.length ; i++) {

System.out.print(a[a.length-i-1]);

}

System.out.println();

}

else {

System.out.println("There is no error in the received data.");

}

System.out.println("Original data sent was:");

power = parity\_count-1;

for(int i=a.length ; i > 0 ; i--) {

if(Math.pow(2, power) != i) {

System.out.print(a[i-1]);

}

else {

power--;

}

}

System.out.println();

}

}

**OUTPUT:**

Enter the number of bits for the Hamming data:

7

Enter bit no. 7:

1

Enter bit no. 6:

0

Enter bit no. 5:

1

Enter bit no. 4:

0

Enter bit no. 3:

1

Enter bit no. 2:

0

Enter bit no. 1:

1

You entered:

1010101

Generated code is:

10100101111

Enter position of a bit to alter to check for error detection at the receiver end (0 for no error):

5

Sent code is:

10100111111

Error is at location 5.

Corrected code is:

10100101111

Original data sent was:

1010101

**CODE:**

import java.io.\*;

public class Fitting

{

BufferedReader input=new BufferedReader(new InputStreamReader(System.in));

int process[], rprocess[], block[], rblock[], usage[], rusage[];

int p, b, free, used, rfree, rused, c, c1;

public Fitting() throws IOException

{

System.out.println("Enter number of blocks");

b=Integer.parseInt(input.readLine());

System.out.println("Enter number of processes");

p=Integer.parseInt(input.readLine());

process=new int[p];

rprocess=new int[p];

block=new int[b];

rblock=new int[b];

usage=new int[b];

rusage=new int[b];

c=0;

}

void read() throws IOException

{

int i;

System.out.println("Enter block sizes");

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

{

System.out.print("Block "+(i+1)+" : ");

rblock[i]=Integer.parseInt(input.readLine());

}

System.out.println("Enter block usage scenario");

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

{

System.out.println("Is block "+(i+1)+" used (1) or free (0)?");

rusage[i]=Integer.parseInt(input.readLine());

if(rusage[i]==1)

{

rused=rused+rblock[i];

c1++;

}

else

rfree=rfree+rblock[i];

}

System.out.println("Enter process demand");

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

{

System.out.print("Process "+(i+1)+" : ");

rprocess[i]=Integer.parseInt(input.readLine());

}

}

void reset()

{

int i;

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

{

block[i]=rblock[i];

usage[i]=rusage[i];

}

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

process[i]=rprocess[i];

used=rused;

free=rfree;

c=c1;

}

void display()

{

int total;

total=rused+rfree;

System.out.println("Blocks used = "+c);

System.out.println("Total used space = "+used);

System.out.println("Blocks free = "+(b-c));

System.out.println("Total free space = "+(total-used));

}

void f\_fit()

{

int i,j;

for(i=0;i<p;i++) //Processes.

for(j=0;j<b;j++) //Blocks.

{

if(process[i]<=block[j]&&usage[j]==0)

{

usage[j]=1;

used=used+block[j];

c++;

System.out.println("Process "+(i+1)+" is in block "+(j+1));

break;

}

}

}

void b\_fit()

{

int i,j,size,best;

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

{

size=32967;

best=-1;

for(j=0;j<b;j++)

{

if(process[i]<=block[j]&&usage[j]==0&&(block[j]-process[i])<size)

{

size=block[j]-process[i];

best=j;

}

}

if(size<32967&&best!=-1) //Ensuring a best fit.

{

usage[best]=1;

used=used+block[best];

c++;

System.out.println("Process "+(i+1)+" is in block "+(best+1));

}

}

}

public static void main(String[] args) throws IOException

{

BufferedReader input=new BufferedReader(new InputStreamReader(System.in));

int option;

String choice;

Fitting f=new Fitting();

f.read();

do

{

f.reset();

System.out.println("Menu");

System.out.println("1. First fit");

System.out.println("2. Best fit");

System.out.println("3. Block information");

System.out.println("Enter option");

option=Integer.parseInt(input.readLine());

switch(option)

{

case 1: f.f\_fit();

break;

case 2: f.b\_fit();

break;

case 3: f.display();

break;

default: System.out.println("Invalid input");

}

f.display();

System.out.println("Press Y to continue");

choice=input.readLine();

}

while(choice.compareToIgnoreCase("y")==0);

}

}

**OUTPUT:**

Enter number of blocks

5

Enter number of processes

5

Enter block sizes

Block 1 : 500

Block 2 : 400

Block 3 : 300

Block 4 : 200

Block 5 : 100

Enter block usage scenario

Is block 1 used (1) or free (0)?

0

Is block 2 used (1) or free (0)?

0

Is block 3 used (1) or free (0)?

0

Is block 4 used (1) or free (0)?

0

Is block 5 used (1) or free (0)?

0

Enter process demand

Process 1 : 100

Process 2 : 350

Process 3 : 400

Process 4 : 150

Process 5 : 200

Menu

1. First fit

2. Best fit

3. Block information

Enter option

1

Process 1 is in block 1

Process 2 is in block 2

Process 4 is in block 3

Process 5 is in block 4

Blocks used = 4

Total used space = 1400

Blocks free = 1

Total free space = 100

Press Y to continue

Y

Menu

1. First fit

2. Best fit

3. Block information

Enter option

2

Process 1 is in block 5

Process 2 is in block 2

Process 3 is in block 1

Process 4 is in block 4

Process 5 is in block 3

Blocks used = 5

Total used space = 1500

Blocks free = 0

Total free space = 0

Press Y to continue

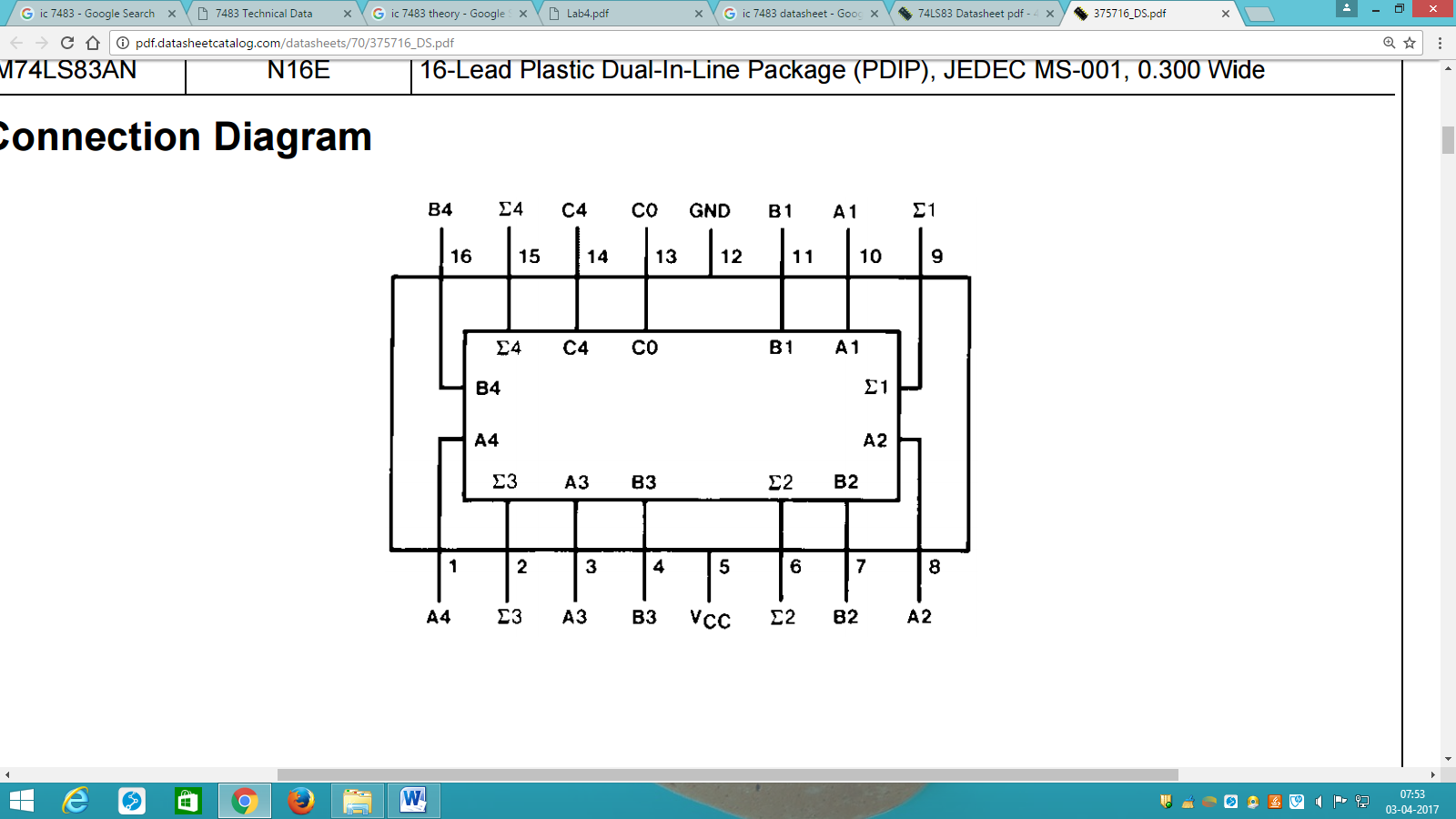
N

**FULL ADDER - IC 7483**

**AIM:** Case Study of IC 7483 Full Adder

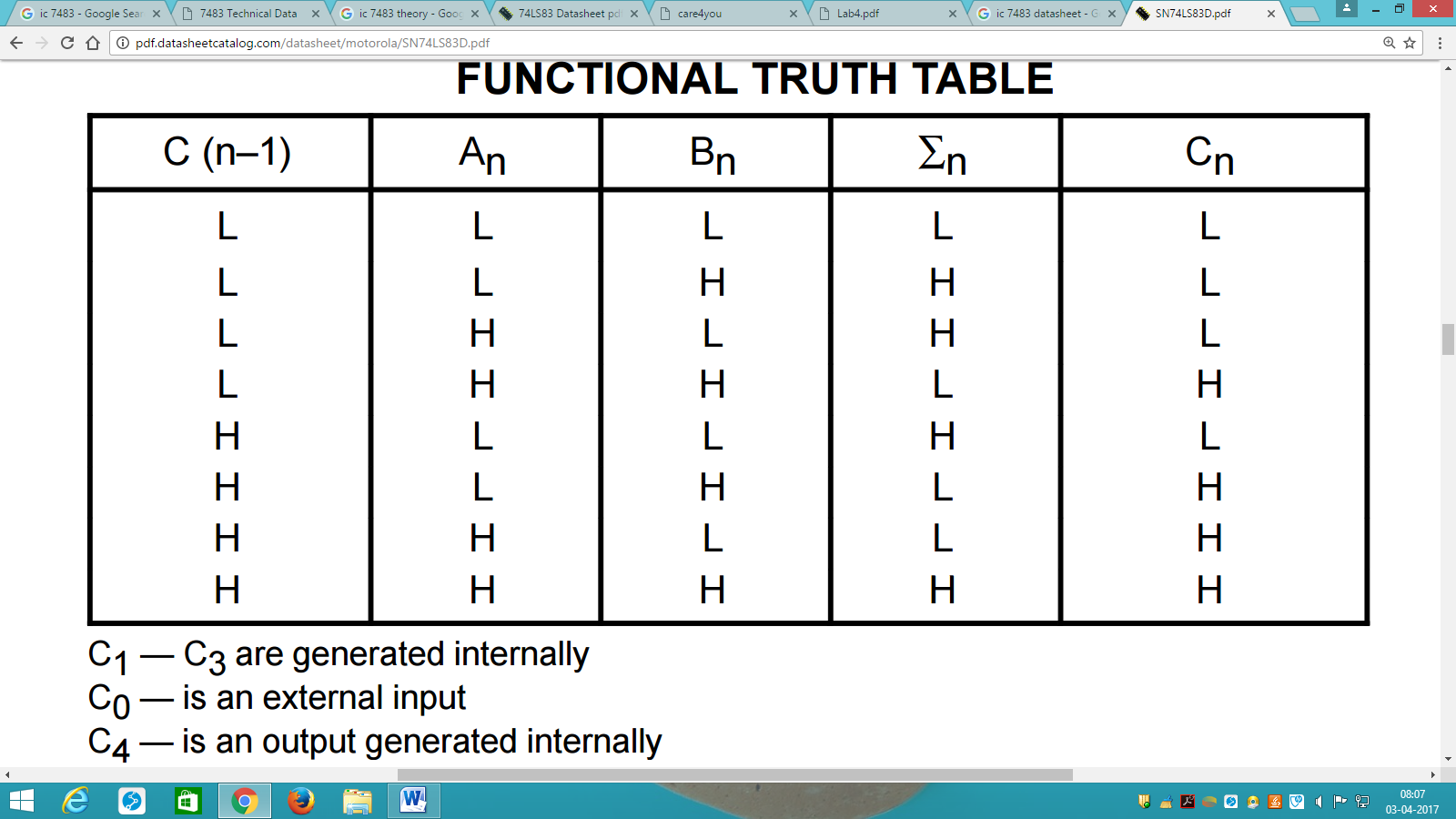
**THEORY:**

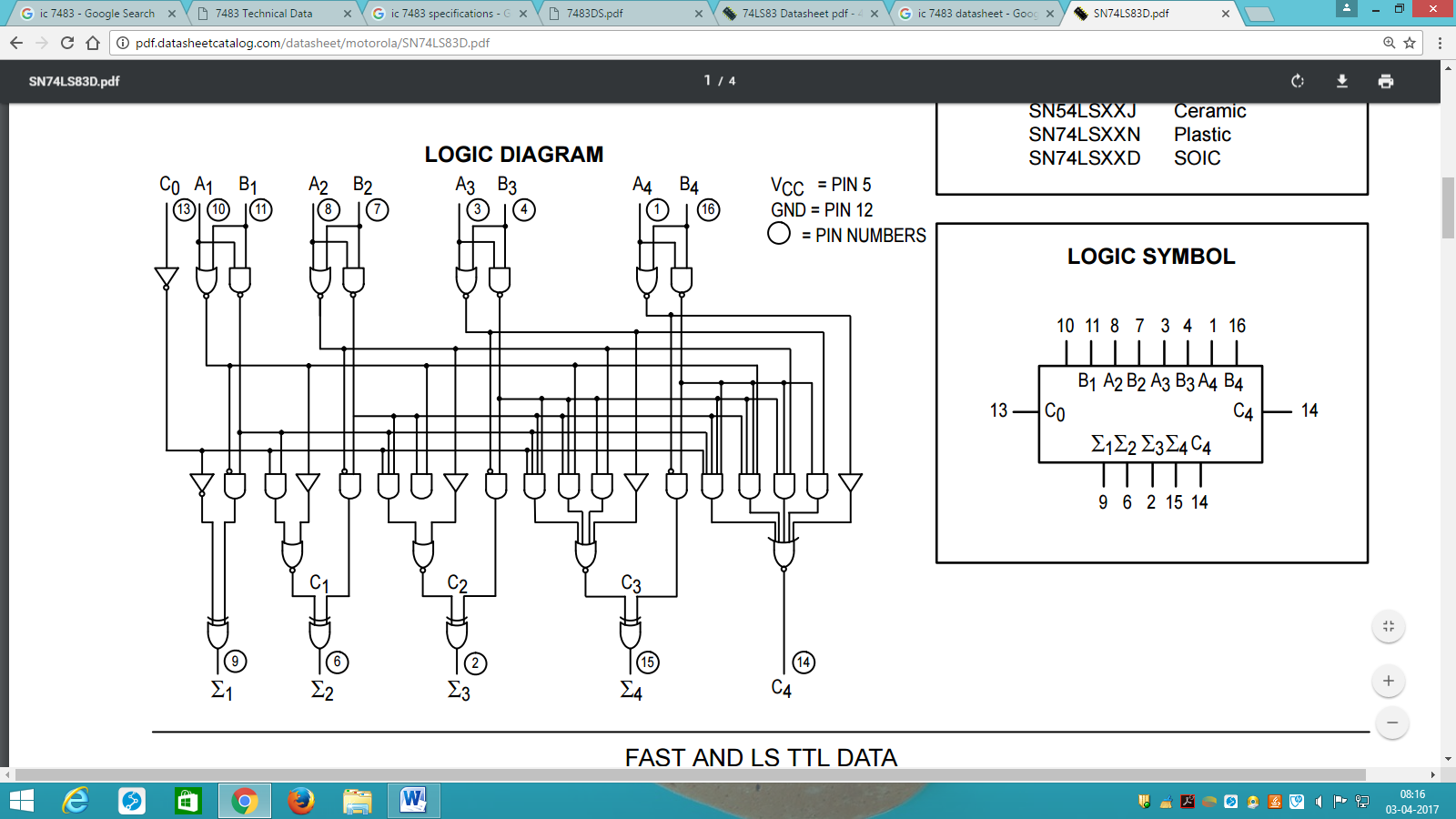
|  |  |
| --- | --- |
| PIN NAMES | DESCRIPTION |
| A1–A4 | Operand A Inputs |
| B1–B4 | Operand B Inputs |
| C0 | Carry Input |
| Σ1–Σ4 | Sum Outputs (Note b) |
| C4 | Carry Output (Note b) |

IC 7483 full adder performs the addition of two 4-bit binary numbers. The sum (∑) outputs are provided for each bit and the resultant carry (C4) is obtained from the fourth bit. It features full internal look ahead across all four bits. This provides the system designer with partial look ahead performance at the economy and reduced package count of a ripple-carry implementation. The adder logic, including the carry, is implemented in its true form meaning that the end-around carry can be accomplished without the need for logic or level inversion.

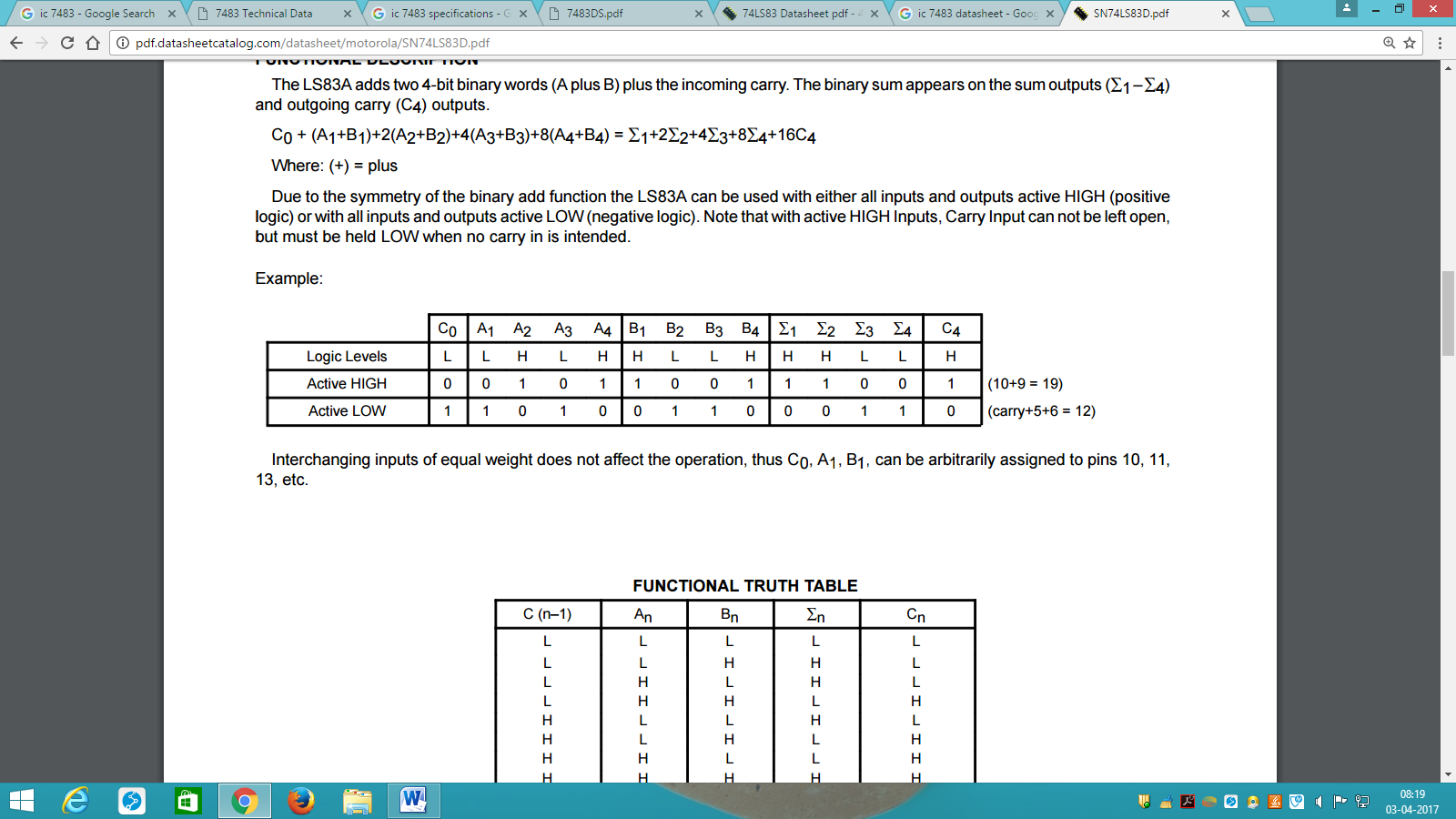
The features of IC 7483 full adder are:

* Full-carry look-ahead across the four bits
* Systems achieve partial look-ahead performance with the economy of ripple carry
* Typical add times Two 8-bit words 25 ns Two 16-bit words 45 ns
* Typical power dissipation per 4-bit adder 95 mW



****

**FUNCTIONAL DESCRIPTION:** The LS83A adds two 4-bit binary words (A plus B) plus the incoming carry. The binary sum appears on the sum outputs (∑1–∑4) and outgoing carry (C4) outputs. C0 + (A1+B1)+2(A2+B2)+4(A3+B3)+8(A4+B4) = ∑1+2∑2+4∑3+8∑4+16C4 Where: (+) = plus Due to the symmetry of the binary add function the LS83A can be used with either all inputs and outputs active HIGH (positive logic) or with all inputs and outputs active LOW (negative logic). Note that with active HIGH Inputs, Carry Input cannot be left open, but must be held LOW when no carry in is intended.

**EXAMPLE:**

**CONCLUSION:** Thus we have successfully studied IC 7483 full adder.

**AIM:** To study the pin configuration and working of ALU IC 74181.

**THEORY:**

The SN54/74LS181 is a 4-bit Arithmetic Logic Unit (ALU) which can perform all the possible 16 logic, operations on two variables and a variety of arithmetic operations.

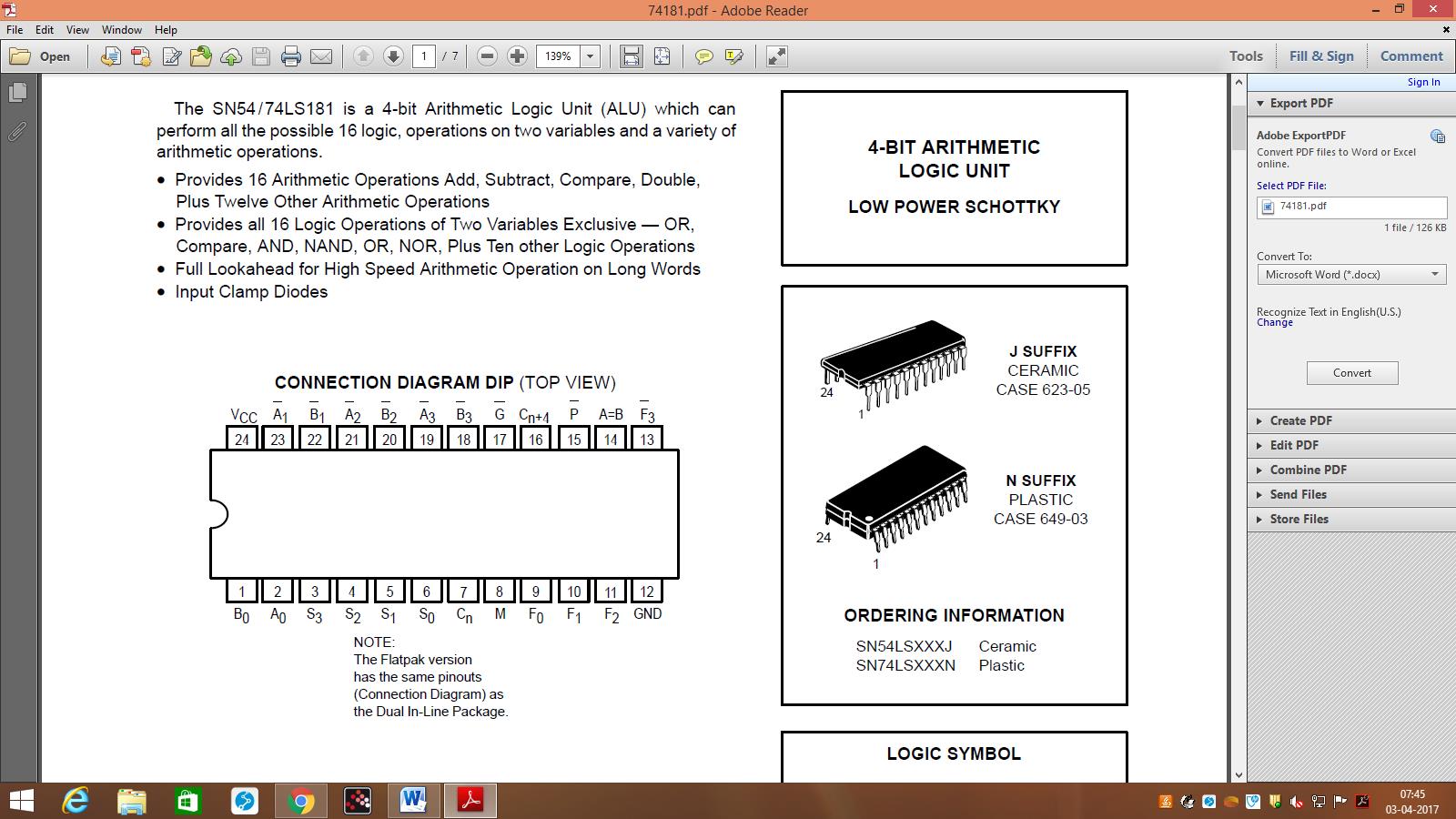
• Provides 16 Arithmetic Operations Add, Subtract, Compare, Double, Plus Twelve Other Arithmetic Operations

• Provides all 16 Logic Operations of Two Variables Exclusive — OR, Compare, AND, NAND, OR, NOR, Plus ten other Logic Operations

• Full Lookahead for High Speed Arithmetic Operation on Long Words

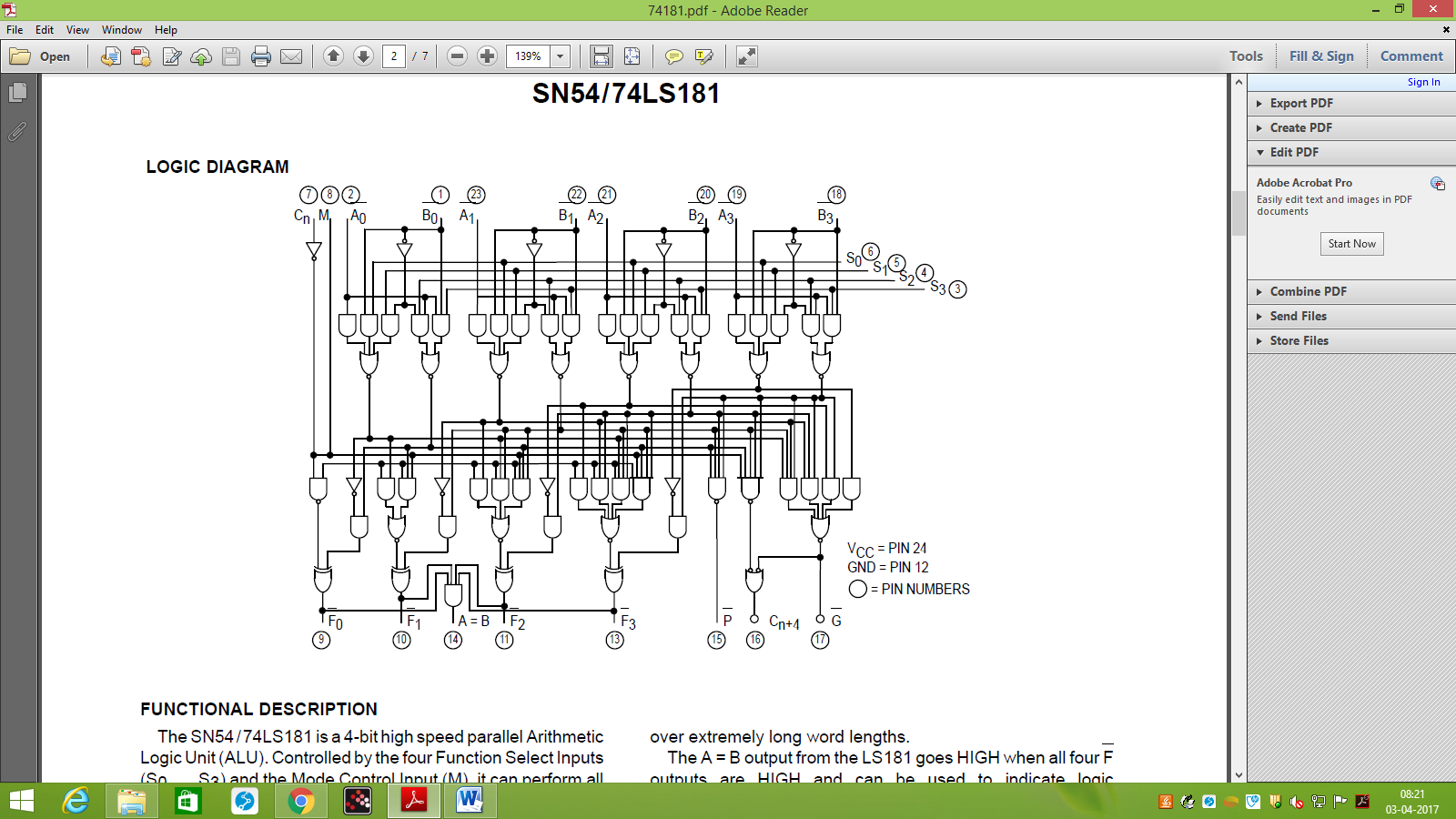
• Input Clamp Diodes

**PIN DIAGRAM:** **PIN DESCRIPTION:**

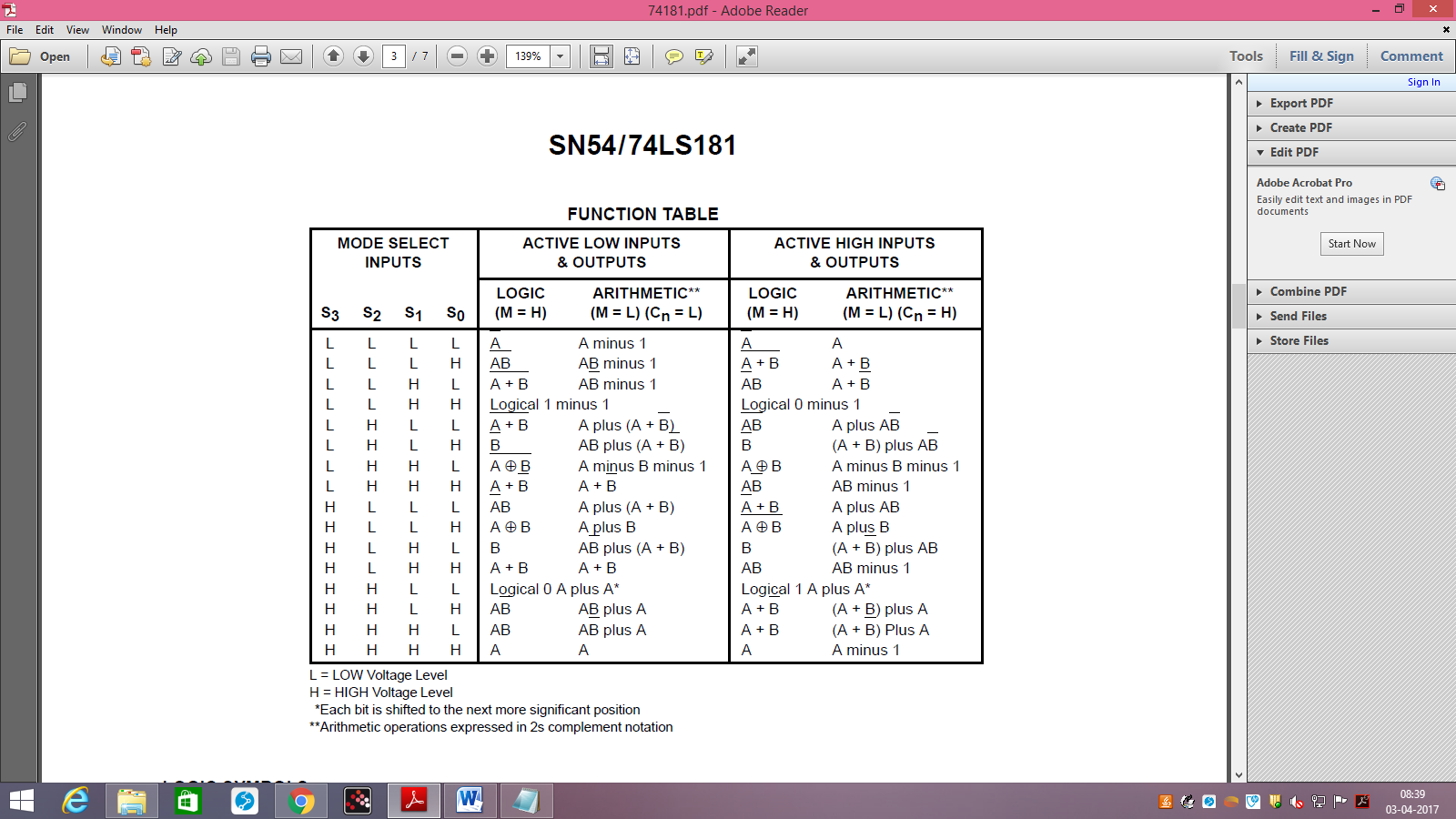


|  |  |
| --- | --- |
| PIN NAMES | DESCRIPTION |
| A0–A3, B0–B3 | Operand (Active LOW) Inputs |
| S0–S3 | Function — Select Inputs |
| M | Mode Control Input |
| Cn | Carry Input |
| F0–F3 | Function (Active LOW) Outputs |
| A = B | Comparator Output |
| G | Carry Generator (Active LOW) Output |
| P | Carry Propagate (Active LOW) Output |
| Cn+4 | Carry Output |

**LOGIC DIAGRAM:**



**FUNCTIONAL DESCRIPTION:**

The SN54/ 74LS181 is a 4-bit high speed parallel Arithmetic Logic Unit (ALU). Controlled by the four Function Select Inputs (S0 . . . S3) and the Mode Control Input (M), it can perform all the 16 possible logic operations or 16 different arithmetic operations on active HIGH or active LOW operands. The Function Table lists these operations. When the Mode Control Input (M) is HIGH, all internal carries are inhibited and the device performs logic operations on the individual bits as listed. When the Mode Control Input is LOW, the carries are enabled and the device performs arithmetic operations on the two 4-bit words. The device incorporates full internal carry lookahead and provides for either ripple carry between devices using the Cn+4 output, or for carry lookahead between packages using the signals P (Carry Propagate) and G (Carry Generate), P and G are not affected by carry in. When speed requirements are not stringent, the LS181 can be used in a simple ripple carry mode by connecting the Carry Output (Cn+4) signal to the Carry Input (Cn) of the next unit. For high speed operation the LS181 is used in conjunction with the 9342 or 93S42 carry lookahead circuit. One carry lookahead package is required for each group of the four LS181 devices. Carry lookahead can be provided at various levels and offers high speed capability over extremely long word lengths. The A = B output from the LS181 goes HIGH when all four F outputs are HIGH and can be used to indicate logic equivalence over four bits when the unit is in the subtract mode. The A = B output is open collector and can be wired-AND with other A = B outputs to give a comparison for more than four bits. The A = B signal can also be used with the Cn+4 signal to indicate A>B and A<B. The Function Table lists the arithmetic operations that are performed without a carry in. An incoming carry adds a one to each operation. Thus, select code LHHL generates A minus B minus 1 (2s complement notation) without a carry in and generates A minus B when a carry is applied. Because subtraction is actually performed by complementary addition (1s complement), a carry out means borrow; thus a carry is generated when there is no underflow and no carry is generated when there is underflow. As indicated, the LS181 can be used with either active LOW inputs producing active LOW outputs or with active HIGH inputs producing active HIGH outputs. For either case the table lists the operations that are performed to the operands

labelled inside the logic symbol.

**CONCLUSION:** Thus we successfully studied ALU IC 74181.