**1).CODE:**

#include<iostream>

#include<math.h>

#include<stdio.h>

#include<conio.h>

#include<algorithm>

#define bool int

#define True 1

#define False 0

void get\_binary\_bits()

{

long Num;

bool flag = 0 ;

std::string set ;

while( Num )

{

if( Num & 1 )

{

set += "True" ;

}

else set += "False" ;

flag = 1;

Num >>= 1 ;

}

if( flag ==0) set += "False" ;

std::reverse( set.begin() , set.end() ) ;

}

int new\_bits()

{

int set = 0 ;

long Num;

while(Num)

{

++set ;

Num >>= 1 ;

}

return set ;

}

bool set\_bits( long long Bin\_bits , int place )

{

long m\_value = 1 << place ;

if( Bin\_bits & m\_value ) return true ;

return false ;

}

long long num\_Arbitary(int n\_bits , bool bit= 0)

{

long long r\_numb = 0 , one = 1 , face ;

int i , r ;

if( bit)n\_bits-=2 ;

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

{

r = rand()%10 ;

if(r > 4 )

{

face = one << i ;

r\_numb |= face ;

}

}

if(bit)

{

face = one << ( n\_bits - 2 ) ;

r\_numb |= face ;

}

return r\_numb ;

}

long long change\_bits( long long Num , int place )

{

long m\_value = 1 << place ; m\_value = ~m\_value ;

Num &= m\_value ;

return Num ;

}

int main()

{

long long Numerator , Divisor , Quotient , Remainder , Two\_cmplmnt\_D;

long long m\_value , one = 1 ;

int i , n , repeat\_bits ;

int n\_bits;

double cycle ;

for( n = 16 ; n <= 32 ; n += 2 )

{

cycle = 0 ;

for(repeat\_bits = 0 ; repeat\_bits < 10000 ; ++repeat\_bits )

{

n\_bits = n \* 2 ;

Numerator = num\_Arbitary( n\_bits ) ;

m\_value = one << n\_bits ;

m\_value -= 1 ;

while( Numerator == m\_value || Numerator == 0 )

{

Numerator = num\_Arbitary( n\_bits ) ;

}

Divisor = num\_Arbitary( n , 1 ) ;

Two\_cmplmnt\_D = (~Divisor ) + 1 ;

Two\_cmplmnt\_D &= ((one<<n)-1) ;

Quotient = 0 ;

Remainder = 0 ;

for( i = n\_bits - 1 ; i >= n ; --i )

{

if( set\_bits( Numerator , i ) && set\_bits( Numerator , i - 1 ) )

{

Quotient <<= 1 ;Quotient |= 1 ;

Numerator = change\_bits( Numerator , i ) ; // removing the bits

continue ;

}

if( set\_bits( Numerator , i ) ==0 && set\_bits( Numerator , i - 1 )==0 )

{

Quotient <<= 1 ;

continue ;

}

break ;

}

if( set\_bits( Numerator , i ) )

{

Quotient |= 1 ;

Quotient <<= 1 ;

Numerator += ( Divisor << ( i - n + 1 ) ) ;

++cycle;

}

else

{

Quotient <<= 1 ;

Numerator += ( Two\_cmplmnt\_D << ( i - n + 1 ) ) ;

++cycle;

}

bool tag = 0 ;

for( ; i >= n ; )

{

Numerator &= ( ( one << (i+1) ) - 1 ) ;

tag = 0 ;

if( set\_bits( Numerator , i ) )

{

Quotient <<= 1 ; Quotient |= 1 ;

Numerator = change\_bits( Numerator , i ) ;

tag = 1 ;

}

else

{

Quotient <<= 1 ;

}

--i ;

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

{

if( set\_bits( Numerator , i ) && set\_bits( Numerator , i - 1 ) )

{

Quotient <<= 1 ;Quotient |= 1 ;

Numerator = change\_bits( Numerator , i ) ;

continue ;

}

if( set\_bits( Numerator , i ) ==0 && set\_bits( Numerator , i - 1 )==0 )

{

Quotient <<= 1 ;

continue ;

}

break ;

}

if( tag )

{

Numerator += ( Divisor << (i-n+1) ) ;

++cycle;

}

else

{

Numerator += ( Two\_cmplmnt\_D << (i-n+1) ) ;

++cycle;

}

}

Remainder = Numerator & ((1<<n)-1);

Quotient = Quotient & ((1<<(n+1))-1) ;

}

std::cout<<"\n\n\t 1).Random Value of the Numerator = "<<Numerator<<"\n\t";

std::cout<<" 2).Random Value of the Divisor = "<<Divisor<<"\n\t";

std::cout<<" 3).Number of Bits Given for (n values) = "<<n<<"\n\t";

std::cout<<" 4).Number of Addition/Subtraction cycle Average = "<< cycle/10000 <<"\n\t";

}

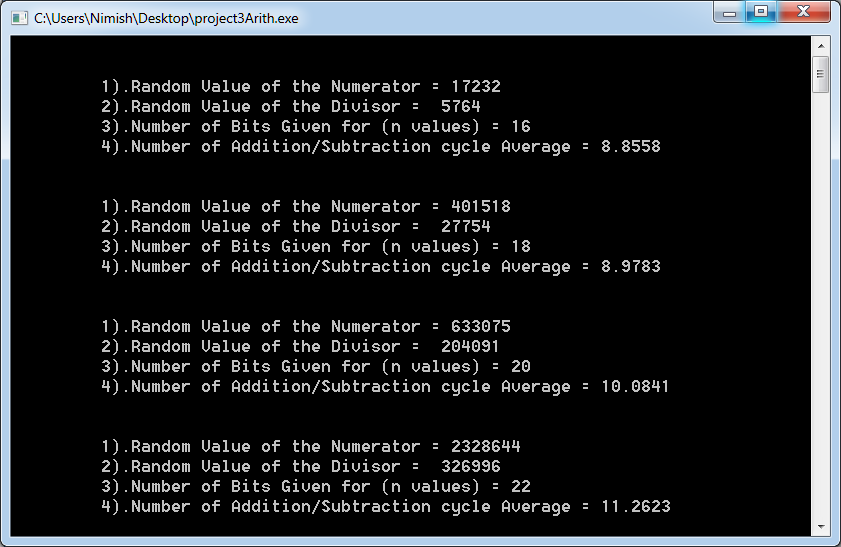
return 0 ;

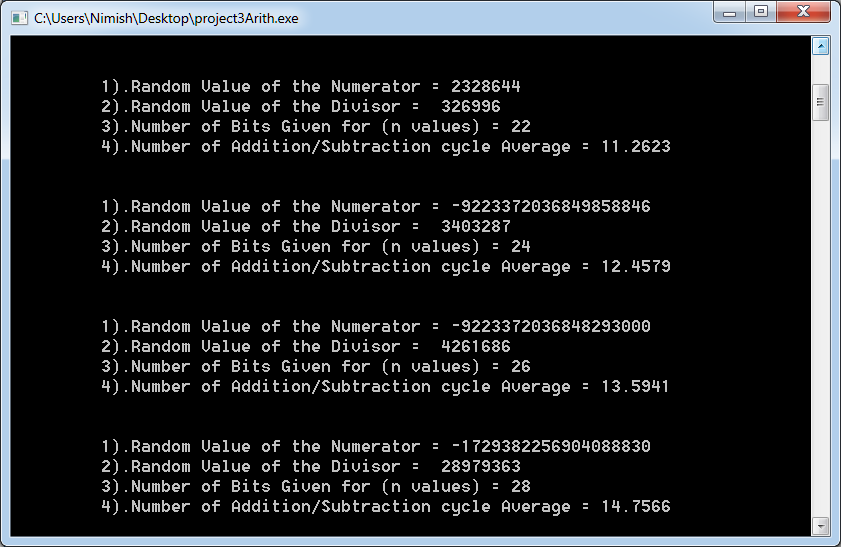
}

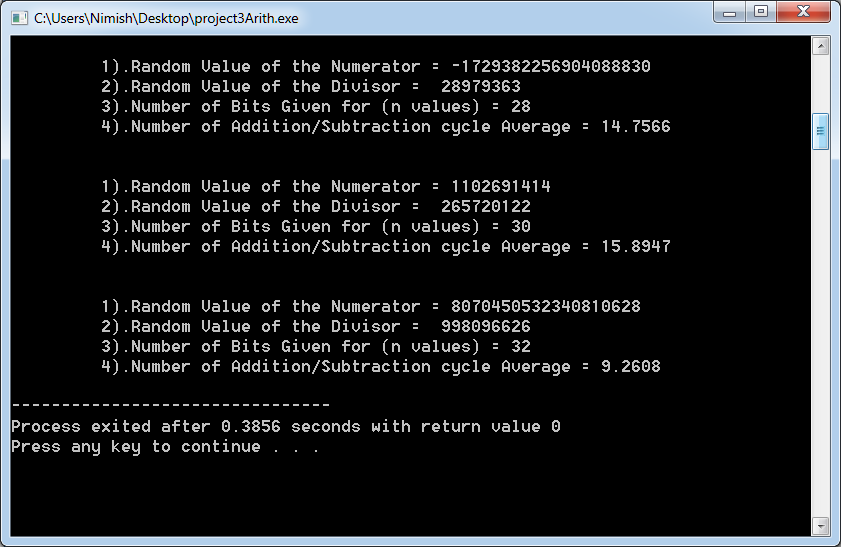
//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end of the file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

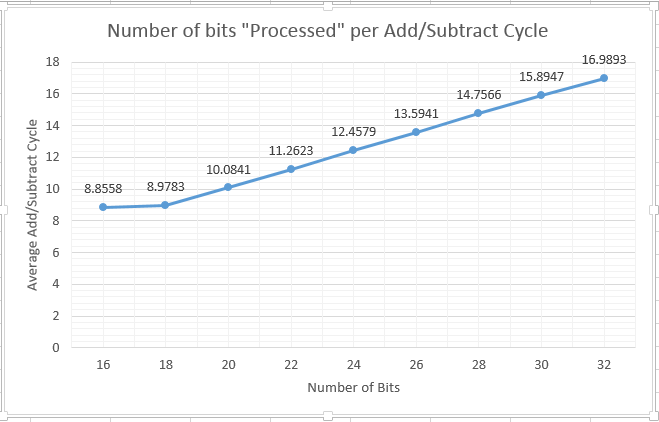
**2).Results:**

|  |  |
| --- | --- |
| **Average Number Of Bits (n)** | **Average Addition/Subtraction Cycle** |
| **16** | **8.8558** |
| **18** | **8.9783** |
| **20** | **10.0841** |
| **22** | **11.2623** |
| **24** | **12.4579** |
| **26** | **13.5941** |
| **28** | **14.7566** |
| **30** | **15.8947** |
| **32** | **16.9893** |





****

****

**3).Step by Step Demonstration:**

Given :

N= 00001001101000001010101010101010

D= 0111101100101010

STEP:1 Remainder1= 0001001101000001

STEP:2 Remainder2 = 0010011010000010

STEP:3 Remainder3 = 0100110100000101

STEP:4 Remainder4 =1100110111011011

STEP:5 Remainder5 =1001101110110110

STEP:6 Remainder6 = 0001101011100000

STEP:7 Remainder7 =0011010111000001

STEP:8 Remainder8 =0110101110000010

STEP:9 Remainder9 =1110110001011000

STEP:10 Remainder10= 1101100010110001

STEP:11 Remainder11= 0101011111011011

STEP:12 Remainder12= 1010111110110110

STEP:13 Remainder13 =0010111011100000

STEP:14 Remainder14 =0101110111000001

STEP:15 Remainder15 =1101111010010111

STEP:16 Remainder16 =1011110100101110

STEP:17 Remainder17 =0011110001011000

STEP:18 Remainder18 =0111100010110001

STEP:19 Remainder19 =1111100110000111

STEP:20 Remainder20 =1111001100001110

STEP:21 Remainder21 =1110011000011101

STEP:22 Remainder22 =1100110000111010

STEP:23 Remainder23 =0100101101100100

STEP:24 Remainder24 =1001011011001001

STEP:25 Remainder25 =0001010111110011

STEP:26 Remainder26 =0010101111100110

Hence the final

Remainder = 0010101111100110 & Quotient = 00001001101011110