Swap all odd and even bits

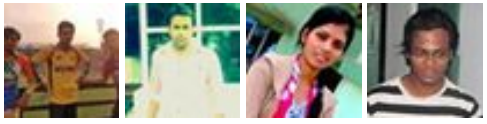
Given an unsigned integer, swap all odd bits with even bits. For example, if the given number is 23 (**0**0**0**1**0**1**1**1), it should be converted to 43 (0**0**1**0**1**0**1**1**). Every even position bit is swapped with adjacent bit on right side (even position bits are highlighted in binary representation of 23), and every odd position bit is swapped with adjacent on left side.

If we take a closer look at the example, we can observe that we basically need to right shift (>>) all even bits (In the above example, even bits of 23 are highlighted) by 1 so that they become odd bits (highlighted in 43), and left shift (<<) all odd bits by 1 so that they become even bits. The following solution is based on this observation. The solution assumes that input number is stored using 32 bits.

Let the input number be x
1) Get all even bits of x by doing bitwise and of x with 0xAAAAAAAA. The number 0xAAAAAAAA is a 32 bit number with all even bits set as 1 and all odd bits as 0.
2) Get all odd bits of x by doing bitwise and of x with 0x55555555. The number 0x55555555 is a 32 bit number with all odd bits set as 1 and all even bits as 0.
3) Right shift all even bits.
4) Left shift all odd bits.
5) Combine new even and odd bits and return.

```
// C program to swap even and odd bits of a given number
#include <stdio.h>

unsigned int swapBits(unsigned int x)
{
    // Get all even bits of x
    unsigned int even_bits = x & 0xAAAAAAAA;
```

```
    // Get all odd bits of x
    unsigned int odd_bits  = x & 0×55555555;

    even_bits >>= 1;  // Right shift even bits
    odd_bits <<= 1;   // Left shift odd bits

    return (even_bits | odd_bits); // Combine even and odd bits
}

// Driver program to test above function
int main()
{
    unsigned int x = 23; // 00010111

    // Output is 43 (00101011)
    printf("%u ", swapBits(x));

    return 0;
}
```

Output:

```
43
```

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Find position of the only set bit
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number
- Find the element that appears once

Writing code in comment? Please use **ideone.com** and share the link here.

28 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Saurabh Verma • 3 months ago

```
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
int main()
{
unsigned int n;
int i=1,t=3;
printf("enter value of n(less than 255)\n");
scanf("%d" &n);
```

```
while(i<8)
{
if((n&t)==pow(2,i))

n=n-pow(2,(i-1));

else if((n&t)<pow(2,i)&&(n&t)!=0) n="n+pow(2,(i-1));" i="i+2;" t="t*4;" }="" print
0;="" }="">
```

Reply • Share ›

rihansh • 3 months ago

i thought it to do this in this manner
actually we have given t hat the number is represented in 32 bit unsigned integ
the bit and the next bit are same then move to the next bit otherwise if 0 1 this
0 this is the case then subtrct the same from the original number and return th

i hope u got me

Reply • Share ›

Ashwini Kumar • 5 months ago

nyc.....

Reply • Share ›

Saurabh Sharma • 7 months ago

```
#include <stdio.h>

int main()
{
int n, l, r, i, temp, x = 1;
scanf("%d", &n);
```

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 31 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

```
l = n << 1; //left shifting
r = n >> 1; //right shifting
for(i = 0; i < 4; i++) // setting all odd places as 0 of the left shiftednumber
{
temp = l & x;
l = l ^ temp;
x = x << 2;
}
x = 2;
for(i = 0; i < 4; i++) //setting all even places as 0 of rightshifted number
{
temp = r & x;
r = r ^ temp;
x = x << 2;
}
n = l | r;
printf("Number n after swapping = %d\n", n);
return 0;
}
```

1 ^ | v • Reply • Share ›

Guest • 7 months ago

```
#include <stdio.h>

int main()
{
int n, l, r, i, temp, x = 1;
scanf("%d", &n);
l = n << 1;
r = n >> 1;
for(i = 0; i < 4; i++)
{
```

```
temp = l & x;
l = l ^ temp;
x = x << 2;
}
x = 2;
for(i = 0; i < 4; i++)
{
temp = r & x;
r = r ^ temp;
x = x << 2;
}
n = l | r;
printf("Number n after swapping = %d\n", n);
return 0;
}
```


Rajdeep • 8 months ago

```
static void swapEvenAndOddBits(int num)
{
int res=0,odd=0,even=0,power=1;
while(num>0)
{
odd=num & 1;
num=num>>1;
even=num &1;
num=num>>1;
res=even*power+res;
power=power*2;
res=odd*power+res;
power=power*2;
}
```

}

System.out.println(res);
}

Reply • Share ›

atiq • 10 months ago

//There is Another Basic approach .... A naive way.
// Negative numbers will work too... without giving wrong impact.

```
#include<iostream>
using namespace std;

long int swapBit(long int n)
{
long int k=1,p=2;
int temp1=0,temp2=0;
int count=0;
while(count<64)
{
temp1=temp2=0;
if(n&k)
temp1=1;
if(p&n)
temp2=1;
if(temp1!=temp2)
```

see more

3 Reply • Share ›

shivali • 11 months ago

0×55555555 how this works

Reply • Share ›

atiq shivali • 10 months ago

see write binary of ox55555555 (mark hexadecimal representation 5=0
it will be like= 0101 0101 0101.....8 times see all odd bits are one...That

```
/* Paste your code here (You may delete these lines if not wri
```

• Reply • Share ›

shivali • 11 months ago

0xAAAAAAAA 0×55555555
how these are worki

• Reply • Share ›

Rohit shivali • 10 months ago

First, we are considering 32 bits. Divide these bits in groups of 4 bits. y
if you remember hexadecimal representation then these groups can re
0 :- 0000
1 :- 0001
so on.. until 10 is reached
10:- 1010 can also be written as A
11 :- B
so on...
15 :- F

Now if you analyze A :- you would see the 1's are already put in the eve
are put in the odd positions 0101
now when you apply bitwise operator 'AND' between NUM(assume nur
you would get the value of NUM in even positions.. Similarly for odd pos
ODD as task. :)

```
/* Paste your code here (You may delete these lines if not wri
```

anupam • a year ago

We can also do it by swapping (moving from right to left) 1st ,2nd bit then 3rd,

anukul • a year ago

Alternate method for the problem :--

```
 #include<stdio.h>
#include<conio.h>

void main()
{
int num,temp,mask=3,count=1;
clrscr();
printf("ENter number :\n");
scanf("%d",&num);

        while(count<=8)
        {
        temp=num & mask;

                if(temp%3==0 || temp==0)
                num=num;
```

see more

vhajela anukul • 11 months ago

Easy and clean!!!

```
/* Paste your code here (You may delete these lines if not wri
```

Anukul Mohil • a year ago

```
#include<stdio.h>
#include<conio.h>

void main()
{
int num,temp,mask=3,count=1;
clrscr();
printf("ENter number :\n");
scanf("%d",&num);

        while(count<=8)//assuming your processor takes 16 bit for a nu
        {
        temp=num & mask;

                if(temp%3==0 || temp==0)
                num=num;
                else
```

see more

rka143 • a year ago

i think there is problem in above logic because when MSB of the number is se
let assume number is 101001001100...
then & with 0xAAAAAAAA will give: 101000001000... and right shift: 11010000

and & with 0x55555555: 000001000100...

and left shift wll give: 000010001000...

and OR operation will give:

110100000100... | 000010001000... == 110110001100 but the answer should
all other are correct except MSB bit in output

herpus rka143 • 6 months ago

you're doing a signed right shift, and he is using unsigned integers

abhishek08aug • a year ago

```
#include<stdio.h>

int swap_even_odd_bits(int num) {
    int even_bits = num&0xAAAAAAAA;
    even_bits = even_bits>>1;

    int odd_bits = num&0x55555555;
    odd_bits = odd_bits<<1;
    return even_bits|odd_bits;
}

int main(){
    unsigned int x = 23; // 00010111
    printf("%u ", swap_even_odd_bits(x));
    return 0;
}
```

moonlight • a year ago

i guess u just should swap the words even with odd in your explanation and cc

Reply • Share ›

Pollob • a year ago

Cool Technique ... :D

Reply • Share ›

Gopichand Godishala • a year ago

It is not getting compiled. Error in line no:10

unsigned int odd_bits = x & 0×55555555 // why????

Reply • Share ›

GeeksforGeeks ➚ Gopichand Godishala • a year ago

Looks like you are using Turbo C compiler. Please use a compiler whe

Reply • Share ›

Gopichand Godishala ➚ GeeksforGeeks • a year ago

No i am using devC++.....

Reply • Share ›

d.mca.iitr ➚ Gopichand Godishala • a year ago

its syntax error in the code ....

correct multiplication symbol to x(alphabate)

i.e

unsigned int odd_bits = x & 0×55555555;

to

unsigned int odd_bits = x & 0x55555555;

it will work :)

```
/* Paste your code here (You may delete these lin
```


Saravanakumaar J • a year ago

Hi,

Why do you shift even bits to right and odd bits to left?

Best regards,
Saravanakumaar J

kartik ➚ Saravanakumaar J • a year ago

If we do it other way (means shift even bits to left and odd bits to right), rightmost bits. Does this answer your question?

Saravanakumaar J ➚ kartik • a year ago

Kartik,

Thanks for responding.

I would like to know why do we require to shift the bits in the firs

atiq ➚ Saravanakumaar J • 10 months ago

If we don't do so.... Bit position will not get reversed(odd

the same result.