```
/*
给定一个正整数，输出它的补数。补数是对该数的二进制表示取反。

注意:

    给定的整数保证在32位带符号整数的范围内。
    你可以假定二进制数不包含前导零位。

示例 1:

输入: 5
输出: 2
解释: 5的二进制表示为101（没有前导零位），其补数为010。所以你需要输出2。

示例 2:

输入: 1
输出: 0
解释: 1的二进制表示为1（没有前导零位），其补数为0。所以你需要输出0。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/number-complement
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 取补用异或运算最直接,这里需要注意的就是从高bit到低bit,遇到第一个为1的bit才开始异或.
- 也就是找到比num大的每位都为1的数,与N进行异或.

方法一:C_Xor

```
int findComplement(int num)
{
    const unsigned int arr[32] =
    {
        0x00000001,0x00000002,0x00000004,0x00000008,
        0x00000010,0x00000020,0x00000040,0x00000080,
        0x00000100,0x00000200,0x00000400,0x00000800,
        0x00001000,0x00002000,0x00004000,0x00008000,
        0x00010000,0x00020000,0x00040000,0x00080000,
        0x00100000,0x00200000,0x00400000,0x00800000,
        0x01000000,0x02000000,0x04000000,0x08000000,
        0x10000000,0x20000000,0x40000000,0x80000000
    };
    int ret_val = 0;
    int i       = 0;
    int flag    = 0;
```

```
    do
    {
        if(num<0)
        {
            break;
        }
        if(num==0)
        {
            ret_val = 1;
            break;
        }

        for(i = 31; i >= 0;i--)
        {
            if(flag==1)
            {
                num ^= arr[i];
            }
            else
            {
                if(num&arr[i])
                {
                    flag = 1;
                    num ^= arr[i];
                }
            }
        }
        ret_val = num;
    }while(0);
    return ret_val;
}

/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了78.57%的用户
内存消耗 :6.8 MB, 在所有 C 提交中击败了73.17%的用户
*/
```

方法二:C(Xor,优化)

```
int findComplement(int num)
{
    const unsigned int arr[32] =
    {
        0x00000001,0x00000002,0x00000004,0x00000008,
        0x00000010,0x00000020,0x00000040,0x00000080,
        0x00000100,0x00000200,0x00000400,0x00000800,
        0x00001000,0x00002000,0x00004000,0x00008000,
        0x00010000,0x00020000,0x00040000,0x00080000,
        0x00100000,0x00200000,0x00400000,0x00800000,
```

```
        0x01000000,0x02000000,0x04000000,0x08000000,
        0x10000000,0x20000000,0x40000000,0x80000000
    };
    int ret_val = 0;
    int i       = 0;
    int flag    = 0;
    int mask    = -1;
    do
    {
        if(num<0)
        {
            break;
        }
        if(num==0)
        {
            ret_val = 1;
            break;
        }

        for(i = 31; i >= 0;i--)
        {
            if(num&arr[i])
            {
                break;
            }
            else
            {
                mask ^= arr[i];
            }
        }
        ret_val = num^mask;
    }while(0);
    return ret_val;
}
```

AlimyBreak
2019.08.02