338. Counting Bits

Given a non negative integer number **num**. For every numbers **i** in the range **0 ≤ i ≤ num** calculate the number of 1's in their binary representation and return them as an array.

Example:

For num = 5 you should return [0,1,1,2,1,2].

Follow up:

- It is very easy to come up with a solution with run time **O(n*sizeof(integer))**. But can you do it in linear time **O(n)** /possibly in a single pass?
- Space complexity should be **O(n)**.
- Can you do it like a boss? Do it without using any builtin function like **__builtin_popcount** in c++ or in any other language.

```
//swift
//author : zzw
class Solution {
    func countBits(num: Int) -> [Int] {
        var nBits:[Int] = [Int](count:num+1,repeatedValue:0);
        var offset:Int = 0;
        var nextMultipleOfTwo = 1;
        for (var i=1;i<=num;i++)
        {
            if i==nextMultipleOfTwo{
                nBits[i]=1;
                offset = nextMultipleOfTwo;
                nextMultipleOfTwo = nextMultipleOfTwo * 2;

            }else{
                nBits[i] = nBits[i-offset]+1;
            }
        }
        return nBits;
    }
}
```