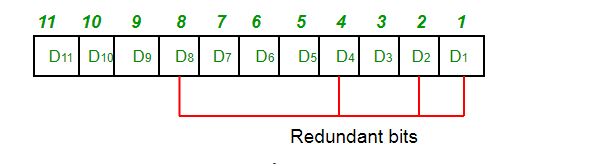
Hamming Codes

The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

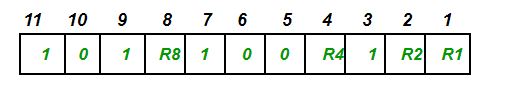
1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.  
   **a.** Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant  
   position (1, 3, 5, 7, 9, 11, etc).  
   **b.** Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from  
   the least significant bit (2, 3, 6, 7, 10, 11, etc).  
   **c.** Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from  
   the least significant bit (4–7, 12–15, 20–23, etc).  
   **d.** Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from  
   the least significant bit bits (8–15, 24–31, 40–47, etc).
5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is  
   odd.
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

**Determining the position of redundant bits –**  
These redundancy bits are placed at the positions which correspond to the power of 2.  
As in the above example:

1. The number of data bits = 7
2. The number of redundant bits = 4
3. The total number of bits = 11
4. The redundant bits are placed at positions corresponding to power of 2- 1, 2, 4, and 8



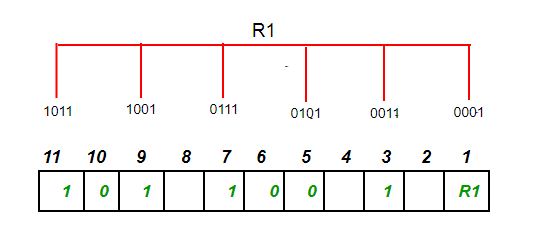
Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



**Determining the Parity bits –**

R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position.

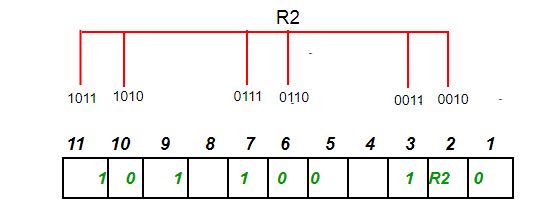
R1: bits 1, 3, 5, 7, 9, 11



To find the redundant bit R1, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit’s value) = 0

R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit.

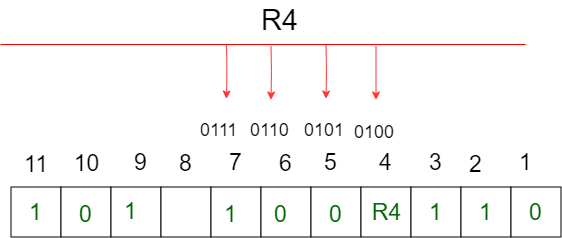
R2: bits 2,3,6,7,10,11



To find the redundant bit R2, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R2 is odd the value of R2(parity bit’s value)=1

R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit.

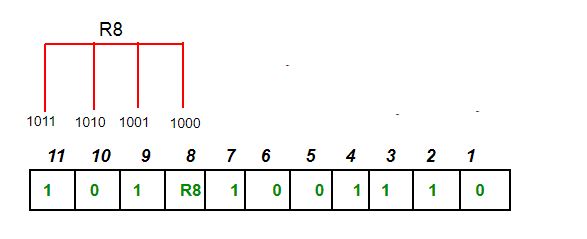
R4: bits 4, 5, 6, 7



To find the redundant bit R4, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R4 is odd the value of R4(parity bit’s value) = 1

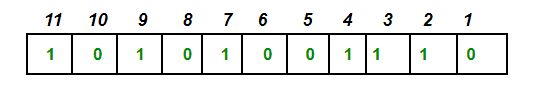
R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit.

R8: bit 8,9,10,11

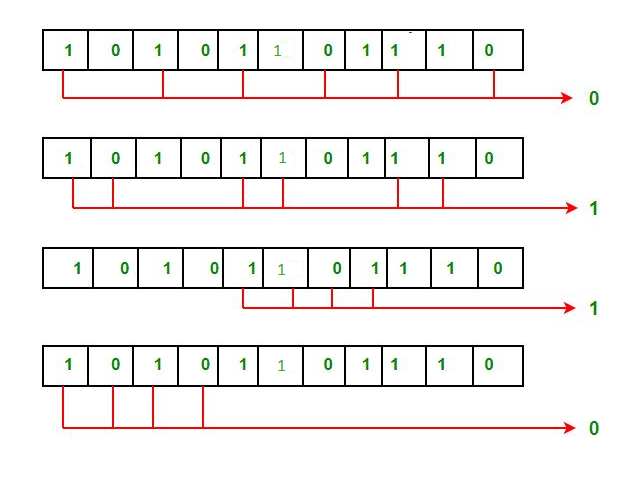


To find the redundant bit R8, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R8 is an even number the value of R8(parity bit’s value)=0.

Thus, the data transferred is:



**Error detection and correction –**  
Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number



The bits give the binary number as 0110 whose decimal representation is 6. Thus, the bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.