

No E-Mail submissions will be accepted.  
Submission formats and file naming:

File name : Pts\_firstName\_lastName\_lab\_7

File format: pdf or MS Word format

e.g. Pts\_Donald\_Trump\_lab\_7.pdf

Reading materials

Use the following link and write a one page summary about the movie.

|  |
| --- |
| **Parity Generator and Parity Checker Explained**  <https://youtu.be/c8qAti1zBVQ>    This video explains how parity generators and parity checkers help detect errors in data transmission. When data is sent between computers or devices, small mistakes can occur due to interference or noise. To catch these errors, an extra bit, called the parity bit, is added to the data. There are two types of parity: even parity, where the total number of 1s ,including the parity bit , must be even, and odd parity, where the total number of 1s must be odd. When the receiver gets the data, it checks whether the number of 1s still matches the expected parity. If it doesn’t, an error is detected, alerting the system that something messed up.  The parity generator circuit creates the parity bit based on the data bits. The video demonstrates how a 3-bit even parity generator works. It takes three data bits, A, B, and C and determines whether the total number of 1s is even or odd. If the number of 1s is even, the parity bit is 0, and if the number of 1s is odd, the parity bit is 1. The circuit for this is built using XOR gates, since an XOR operation outputs 1 when the number of 1s is odd. The formula for this even parity bit is A XOR B XOR C. For odd parity, the logic is just the opposite, so the output is the inverse of the even parity bit and can be created by adding a NOT gate.  Once the data is sent along with the parity bit, the parity checker at the receiving end verifies if the data has been altered. It does this by checking the total number of 1s, including the parity bit. If the parity check does not match the expected value, it means an error has occurred. However, while this method is effective at detecting single-bit errors, it cannot correct them. Also, if two bits flip, the parity remains unchanged, and the error goes undetected. This is a limitation of simple parity checking.  To simplify parity operations, special integrated circuits or ICs like the 74180 IC are available. These chips can automatically generate and check parity for multiple bits at once. Parity checking is widely used in computer systems, networking, and storage devices as a basic form of error detection, ensuring data integrity in many applications. |

1) The word “Covid19” is given.

(a) Represent in an 8-bit ASCII code.

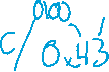
Covid19 => 01000011 01101111 01110110 01101001 01100100 00110001 00111001

Or 0x43 0x6F 0x76 0x69 0x64 0x31 0x39

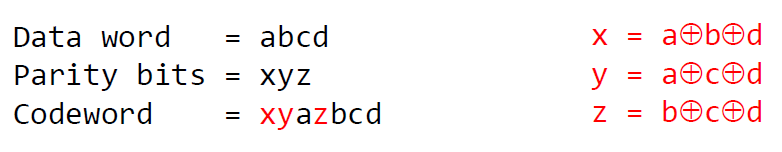
|  |  |  |  |
| --- | --- | --- | --- |
| Letter | ASCII (dec) | ASCII (bin) | ASCII |
| C | 67 | 01000011 | 0x43 |
| o | 111 | 01101111 | 0x6F |
| v | 118 | 01110110 | 0x76 |
| i | 105 | 01101001 | 0x69 |
| d | 100 | 01100100 | 0x64 |
| 1 | 49 | 00110001 | 0x31 |
| 9 | 57 | 00111001 | 0x39 |

(b) Add even parity bit to each character (from left) and represent each character in hex,

|  |  |  |  |
| --- | --- | --- | --- |
|  | ASCII(Hex) | Code word = Parity bit + data  Binary format | Code word = parity bit + data  Hex format |
| **K** | 0x4B | 001001011 | 0x04B |
| **C** | 0x43 |  |  |
| **o** | 0x6F |  |  |
| **v** | 0x76 |  |  |
| **i** | 0x69 |  |  |
| **d** | 0x64 |  |  |
| **1** | 0x31 |  |  |
| **9** | 0x39 |  |  |



2) Use the given formula below to obtain the 7‐bit Hamming codeword for the following 4‐bit data words:





1. 1010



1. 1100



1. 1110



3) Which of the following 7-bit Hamming codes is corrupted?



1. 1011011



1. 0011001



1. 1010000



4) Obtain the 12‐bit Hamming code word for the following 8‐bit data word.



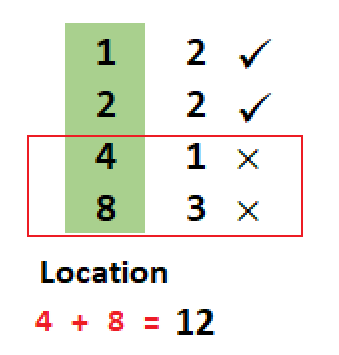
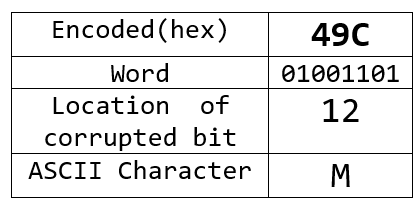
11011011



5) The following string of hex digits encodes extended ASCII characters in an even parity Hamming code: 0D3 DD3 0F2 5C1 1C5 CE3. Decode this string and write down the characters that are encoded (**8 data bits + 4 check bits = 12 bits**).

e.g. **49C**

010010011100 ⇒ 010010011101 ⇒ 01001101 0x4D (77) ⇒ M

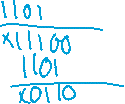




|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Encoded(hex) | **0D3** | **DD3** | **0F2** | **5C1** | **1C5** | **CE3** |
| Word |  |  |  |  |  |  |
| Location of corrupted bit |  |  |  |  |  |  |
| ASCII Character |  |  |  |  |  |  |



6) Encode the data bit sequence 100110000 using the generator 1101 and give the codeword.



8 bits:

Example: A (0x41) = 01000001

