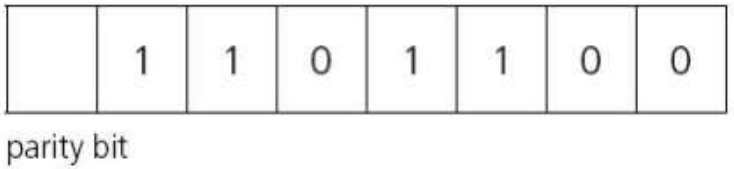
ERROR CHECKING METHODS

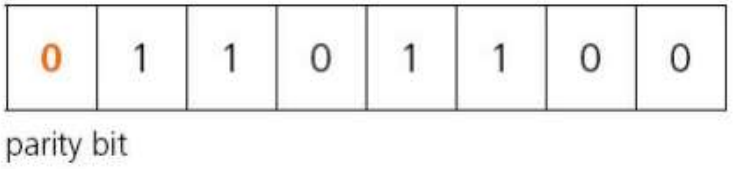
- Errors in data transmission could happen. Therefore, it is important to detect errors
- Error checking methods in IGCSE level:
  - Parity checking
  - ARQ (Automatic Repeat Request)
  - Checksum
  - Echo checking

ERROR CHECKING METHODS

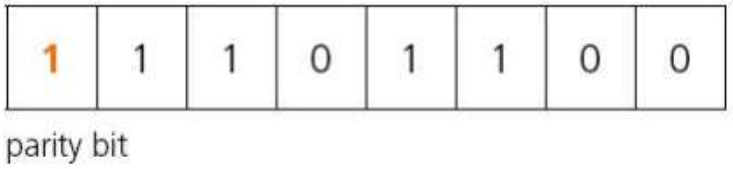
- Parity checking
  - The left most significant bit is parity bit
  - Put 0 or 1 to make the number of 1s to be odd or even (depend on the protocol)
  - There are two systems: even parity and odd parity
    - even parity : one bit to make data to have even numbers of 1s
    - odd parity : one bit to make data to have odd numbers of 1s



either (even parity)

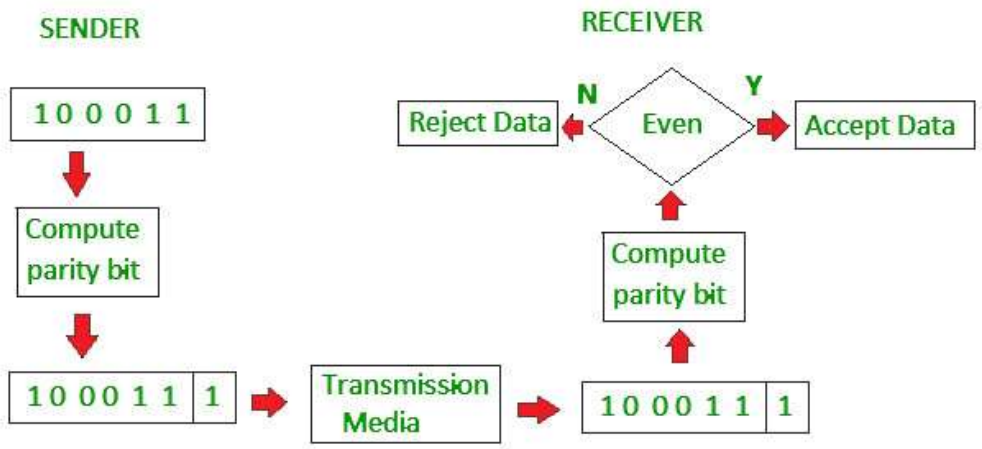


or (odd parity)



ERROR CHECKING METHODS

- How does this work?
  - Sender and receiver must agree what parity bit(odd or even) to be used
  - Sender counts number of 1s and then put 0 or 1 in the parity bit
  - The data is transferred to the receiver
  - Receiver counts number of 1s and check if it is even or odd. If the number of 1s is the same as sender side, the data has been transferred correctly



ERROR CHECKING METHODS

In this example, nine bytes of data have been transmitted. Agreement has been made that even parity will be used. Another byte, known as the **PARITY BYTE**, has also been sent. This byte consists entirely of the parity bits produced by the vertical parity check. The parity byte also indicates the end of the block of data.

The following table shows how the data arrived at the receiving end:

Table 2.2

	parity bit	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8
byte 1	1	1	1	1	0	1	1	0
byte 2	1	0	0	1	0	1	0	1
byte 3	0	1	1	1	1	1	1	0
byte 4	1	0	0	0	0	0	1	0
byte 5	0	1	1	0	1	0	0	1
byte 6	1	0	0	0	1	0	0	0
byte 7	1	0	1	0	1	1	1	1
byte 8	0	0	0	1	1	0	1	0
byte 9	0	0	0	1	0	0	1	0
parity byte	1	1	0	1	0	0	0	1

ERROR CHECKING METHODS

Scenario that parity check does not work:

- even numbers of error (2, 4, 6, …)
- transposition error

0	1	0	1	1	1	0	0

This byte could reach the destination as:

0	1	**1**	1	1	1	0	**1**

or:

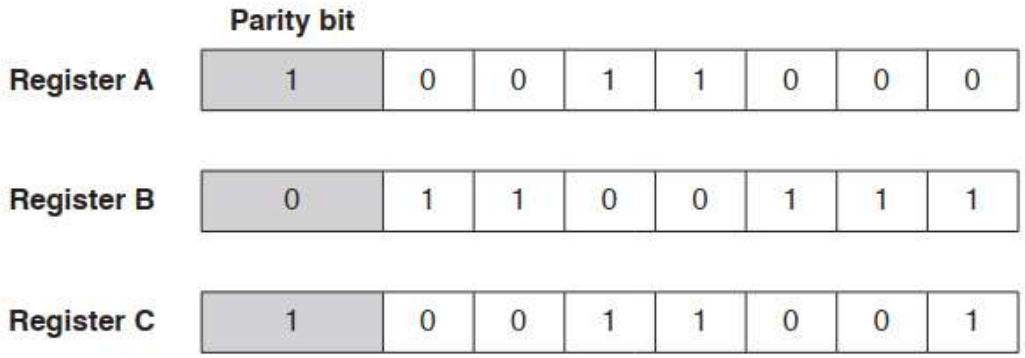
0	1	0	1	**0**	**0**	0	0

or:

0	1	0	1	**0**	1	**1**	0

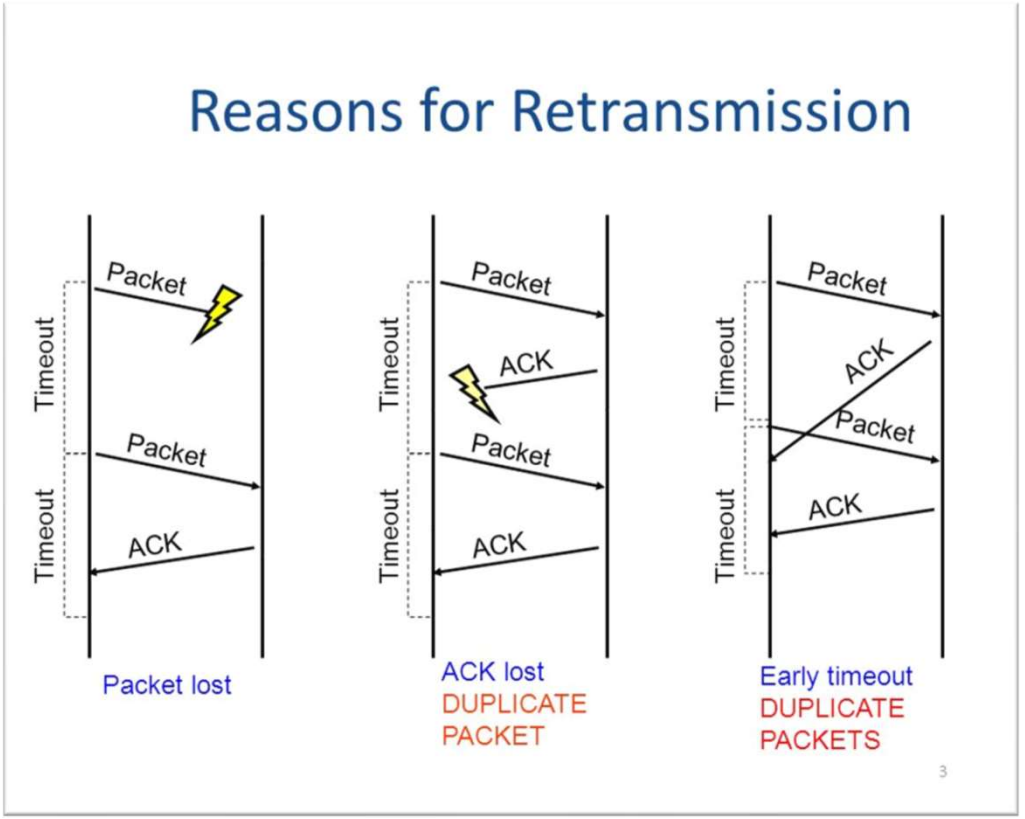
ERROR CHECKING METHODS

- Two bytes have been transmitted correctly. Which register holds the incorrect byte and why it is.



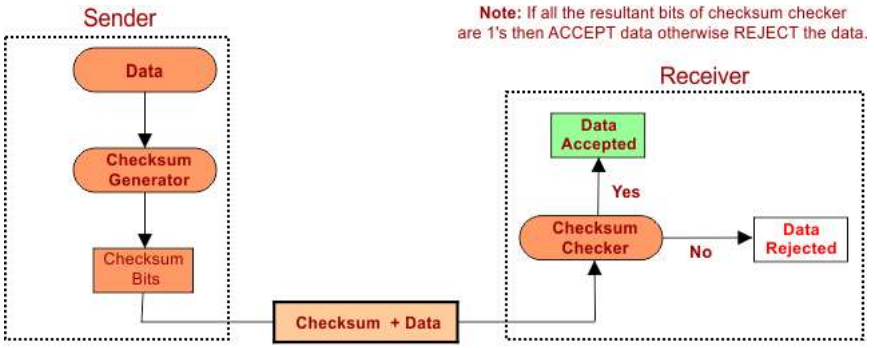
ERROR CHECKING METHODS

- ARQ (Automatic Repeat Request)
  - Sender sends data
  - Sender waits for acknowledge signal
  - If sender doesn't receive acknowledge signal before time out, resend the data again



ERROR CHECKING METHODS

- Check Sum
  - when a data is about to be transmitted, the checksum is calculated from the data
  - the calculation is done using an agreed algorithm (this algorithm has been agreed by sender and receiver)
  - the checksum is then transmitted with the block of data
  - at the receiving end, the checksum is recalculated by the computer using the data (the agreed algorithm is used to find the checksum)
  - the re-calculated checksum is then compared to the checksum sent with the data block
  - if the two checksums are the same, then no transmission errors have occurred; otherwise a request is made to re-send the block of data.

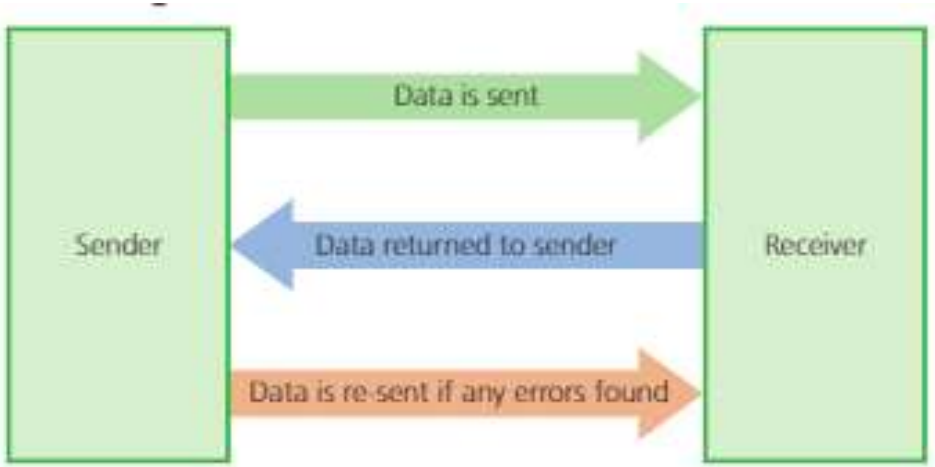


ERROR CHECKING METHODS

- Check sum value can be calculated from totaling ASCII of the word
  - Example, the word “CAT” is needed to be transmitted over the internet
    - C = 67 A = 65 T = 84
    - Total value = 216
    - “CAT” and 216 are sent to the receiver
    - But on the receiver side, “T” is corrupted and change to “R”, so the receiver gets “CAR” instead of “CAT”
    - The receiver recalculate the check sum value again which is 214 because R = 82
    - The sent check value and recalculated value are not the same, it indicates the data is corrupted during transfer

ERROR CHECKING METHODS

- Echo checking
  - sender sends data to receiver
  - receiver sends data back to receiver
  - sender compares original data with sent back data
  - if same, no error
  - if not same, there is an error but cannot identify
    - error in sending process
    - error in reply process



▲ **Figure 2.14** Echo check diagram

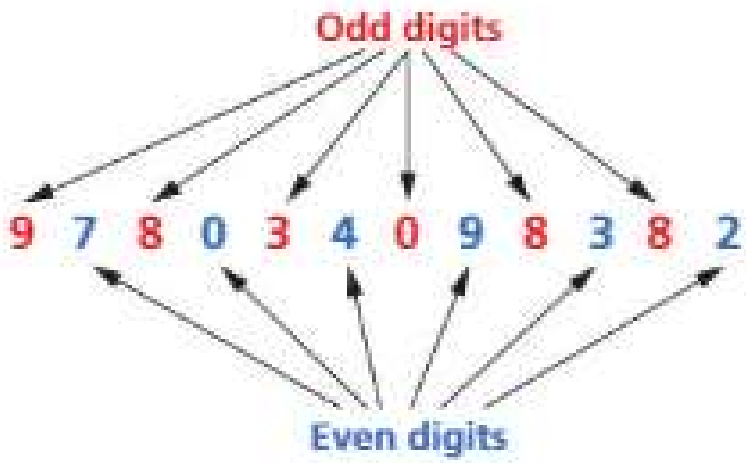
ERROR CHECKING METHODS

- Check digit
  - Found in barcodes, ISBN
  - It can be used to detect error e.g., typing 5037 instead of 5307
  - There are tow common methods which are ISBN13 and Modulo-11

Check digit website

ERROR CHECKING METHODS

- Check digit
  - ISBN 13
  1. add all the odd numbered digits together
  2. add all the even numbered digits together and multiply the result by 3
  3. add the results from 1 and 2 together and divide by 10
  4. take the remainder, if it is zero then use this value, otherwise subtract the remainder from 10 to find the check digit.



▲ **Figure 2.16** ISBN (no check digit)

1 9 + 8 + 3 + 0 + 8 + 8 = 36
2 3 × (7 + 0 + 4 + 9 + 3 + 2) = 75
3 (36 + 75)/10 = 111/10 = 11 remainder 1
4 10 – 1 = 9 the check digit

ERROR CHECKING METHODS

- Check digit
  - Modulo-11
    1. each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3 or 2 starting from the left (weightings start from 8 since the number will become eight-digit when the check digit is added)
    2. the digit is multiplied by its weighting and then each value is added to make a total
    3. the total is divided by 11
    4. the remainder is then subtracted from 11 to find the check digit (note if the remainder is 10 then the check digit 'X' is used).

The example to be used has the following seven-digit number:

1 7-digit number: 4 1 5 6 7 1 0

weighting values: 8 7 6 5 4 3 2

2 sum: (8 × 4) + (7 × 1) + (6 × 5) + (5 × 6) + (4 × 7) + (3 × 1) + (2 × 0)
= 32 + 7 + 30 + 30 + 28 + 3 + 0
total = 130

3 divide total by 11: 130/11 = 11 remainder 9

4 subtract remainder from 11: 11 – 9 = 2 (check digit)

So we end up with the following eight-digit: 4 1 5 6 7 1 0 2