# Worksheet 2 Bits, bytes and binary Answers

**Task 1**

1. Convert the following decimal values into 8-bit binary bytes:
2. 1010  = 00001010
3. 10410  = 01101000
4. 25510  = 11111111
5. A single byte can be used to represent the decimal values 010 to 25510. For values over 25510 bytes can be joined together. In a computer that has a 16-bit bus width, an integer would be stored in two consecutive bytes.

For example, to represent 65410 the two bytes used would be:

|  |  |
| --- | --- |
| **Byte 2** | **Byte 1** |
| **215** | **214** | **213** | **212** | **211** | **210** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** |
| 32,768 | 16,384 | 8,192 | 4,096 | 2,048 | 1,024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** | **0** | **0** | **0** | **1** | **1** | **1** | **0** |

Convert the following decimal values into 2 bytes:

1. 12710  byte 2 = 00000000 and byte 1 = 01111111
2. 318810  byte 2 = 00001100 and byte 1 = 01110100
3. 6553510  byte 2 = 11111111 and byte 1 = 11111111
4. Put the following byte prefixes in order of size from smallest to largest:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **mega** | **gibi** | **kibi** | **tebi** | **kilo** | **giga** | **tera** | **mebi** |

|  |  |  |
| --- | --- | --- |
| **Prefix** | **Symbol** | **Number of bytes** |
| kilo | k | 1,000 |
| kibi | Ki | 1,024 |
| mega | M | 1,000,000 |
| mebi | Mi | 1,048,576 |
| giga | G | 1,000,000,000 |
| gibi | Gi | 1,073,741,824 |
| tera | T | 1,000,000,000,000 |
| tebi | Ti | 1,099,511,627,776 |

# Task 2 Representing characters Answers

1. Using the ‘ASCII codes’ helpsheet, answer the following questions:
2. What is your forename in ASCII?

Answers will vary.

1. Convert the following ASCII sentence to text:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **01000100** | **01001110** | **01000001** | **00100000** | **01110111** | **01100001** |
| D | N | A |  | w | a |
| **01110011** | **00100000** | **01100100** | **01101001** | **01110011** | **01100011** |
| s |  | d | i | s | c |
| **01101111** | **01110110** | **01100101** | **01110010** | **01100101** | **01100100** |
| o | v | e | r | e | d |
| **00100000** | **01101001** | **01101110** | **00100000** | **00110001** | **00111001** |
|  | i | n |  | 1 | 9 |
| **00111000** | **00110100** | **00101110** |  |  |  |
| 8 | 4 | . |  |  |  |

1. Explain why when adding the characters ‘2’ + ‘3’ you don’t get 5:

As the numbers are codes for the characters ‘2’ and ‘3’ their binary values are 50 and 51. Adding these values together does not give the answer 5. If this was written into a computer program the 2 and 3 characters would be treated as strings and the output would be 23.

1. Create a spreadsheet that can convert a word of up to 8 characters into ASCII character codes.



Extend the spreadsheet to convert ASCII binary codes back to regular characters.



# Task 3 Error checking Answers

# Using a barcode on the back of a book, calculate the check digit using the Modulo 10 system.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ISBN** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Weight** | 1 | 3 | 1 | 3 | 1 | 3 | 1 | 3 | 1 | 3 | 1 | 3 |  |
| **Multiplication** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Addition** | Add all the numbers |  |
| **Remainder** | Find the remainder when divided by 10 |  |
| **Subtraction** | Subtract the result from 10 |  |

# The following three bytes are transmitted across a Serial interface using odd parity. Insert the parity bits for each byte that is transmitted.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |  | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |  | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

# The Luhn algorithm was devised as a checksum formula to ensure credit card numbers are valid when manually or automatically entered into a machine.

The steps in the algorithm are as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Credit Card Number | 4 | 3 | 6 | 2 | 6 | 2 | 6 | 8 | 7 | 7 | 4 | 3 | 3 | 1 | 1 | 6 |  |
| Double every other number | **8** |  | **12** |  | **12** |  | **12** |  | **14** |  | **8** |  | **6** |  | **2** |  |  |
| Subtract 9 if number > 9 |  |  | **3** |  | **3** |  | **3** |  | **5** |  |  |  |  |  |  |  |  |
| Find sum of all digits | **8** | **3** | **3** | **2** | **3** | **2** | **3** | **8** | **5** | **7** | **8** | **3** | **6** | **1** | **2** | **6** | **70** |

 If the sum of all digits is a number divisible by 10, the number will be accepted. If not, it is rejected assuming an error in input.

# Would the following credit card number be accepted? Show your working. Yes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Credit Card Number | 4 | 4 | 7 | 4 | 8 | 5 | 2 | 4 | 6 | 6 | 7 | 8 | 5 | 4 | 8 | 5 |  |
| Double every other number | **8** |  | **14** |  | **16** |  | **4** |  | **12** |  | **14** |  | **10** |  | **16** |  |  |
| Subtract 9 if number > 9 |  |  | **5** |  | **7** |  |  |  | **3** |  | **5** |  | **1** |  | **7** |  |  |
| Find sum of all digits | **8** | **4** | **5** | **4** | **7** | **5** | **4** | **4** | **3** | **6** | **5** | **8** | **1** | **4** | **7** | **5** | **80** |