**Data Representation and Boolean Logic**

Part A: Data Representation

**Date of Birth: 19/04/1997**

**Question 1**

19 + 15 = 34

D10 = 34

**Question 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

D2 = 00100010

**Question 3**

4 + 1 = 5

M10 = 5

**Question 4**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

M2 = 00000101

**Question 5**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sign bit | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

D2C = 00100010

As the number is a positive integer, the actual bit pattern does not change. However, rather than being used to represent a value, the first bit is now used to represent the sign of the number. In this case, the sign bit is ‘0’, therefore indicating that the value is positive.

**Question 6**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sign bit | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

M2C = 00000101

As the number is a positive integer, the actual bit pattern does not change. However, rather than being used to represent a value, the first bit is now used to represent the sign of the number. In this case, the sign bit is ‘0’, therefore indicating that the value is positive.

**Question 7**

**Convert D2C to a negative binary integer**

The only way to complete binary subtraction is to make the second binary value negative. This is done using Two’s complement. In this case, I needed to represent the value of D2C as a negative binary integer. To do this, I used the following steps:

Step 1

Starting from the right-most bit, copy each bit until the first bit set to ‘1’ has been copied:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sign bit | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
|  |  |  |  |  |  | 1 | 0 |

Step 2

For each of the remaining bits, flip their value so that every ‘0’ is set to ‘1’ and vice versa:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sign bit | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |

-D2C = 11011110

**Perform the binary subtraction**

Now that I have the negative representation of D2C, the binary subtraction can now be performed. This is done by adding the value of M2C to the negative representation of D2C as shown below:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M2C | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | + |
| -D2C | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| **R2C** | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |  |
| (carry) |  |  | 1 | 1 | 1 |  |  |  |  |

R2C = 11100011

**Question 8**

To convert the 8-bit Two’s complement value into the unassigned 8-bit binary value, I took note of the sign (in this case the sign bit of ‘1’ means that the number is negative), and then followed the same steps as displayed in question 7 in order to obtain the regular binary representation of the number.

Step 1

Starting from the right-most bit, copy each bit until the first bit set to ‘1’ has been copied:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
|  |  |  |  |  |  |  | 1 |

Step 2

For each of the remaining bits, flip their value so that every ‘0’ is set to ‘1’ and vice versa:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

R2 = -00011101

**Question 9**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |

-(24 + 23 + 22 + 20)= -(16 + 8 + 4 + 1)

-(16 + 8 + 4 + 1) = -29

R10 = -29

Part B: Boolean Logic

**Question 1**

Based on my analysis of the scenario, I can conclude that the boolean expression that represents this scenario is as follows:

**Y = (A . B . C) + D**

In the above expression:

* ‘A’ represents ‘Foxglove’
* ‘B’ represents ‘Fuschia’
* ‘C’ represents ‘Dianthus’
* ‘D’ represents ‘Gladioli’
* ‘Y’ represents ‘Compost’

The truth table below summarises every possible combination of the four special-offer plant varieties and shows, based on the boolean expression, whether or not the combination of plants means that the customer earns a free sack of compost.

**Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Foxglove** | **Fuschia** | **Dianthus** |  | **Gladioli** | **Compost?** |
| **A** | **B** | **C** | **(A.B.C)** | **D** | **Y = (A . B . C) + D** |
| 0 | 0 | 0 | **0** | **0** | **0** |
| 1 | 0 | 0 | **0** | **0** | **0** |
| 0 | 1 | 0 | **0** | **0** | **0** |
| 0 | 0 | 1 | **0** | **0** | **0** |
| 1 | 1 | 0 | **0** | **0** | **0** |
| 0 | 1 | 1 | **0** | **0** | **0** |
| 1 | 0 | 1 | **0** | **0** | **0** |
| 1 | 1 | 1 | **1** | **0** | **1** |
| 0 | 0 | 0 | **0** | **1** | **1** |
| 1 | 0 | 0 | **0** | **1** | **1** |
| 0 | 1 | 0 | **0** | **1** | **1** |
| 0 | 0 | 1 | **0** | **1** | **1** |
| 1 | 1 | 0 | **0** | **1** | **1** |
| 0 | 1 | 1 | **0** | **1** | **1** |
| 1 | 0 | 1 | **0** | **1** | **1** |
| 1 | 1 | 1 | **1** | **1** | **1** |

**Question 2**

Below is a simple digital circuit diagram which illustrates the logic of the special offer and therefore could be used to control that part of the greenhouse packing decision making process.

