1(a)	FFF
1(b)	4095
1(c)	0100 0010 1001 0001 1010 0011
1(d)	8A1 70B
1(e)(i)	Microprocessor
1(e)(ii)	A
1(e)(iii)	Light
1(e)(iv)	Any **four** from: • The **disk** is spun • A laser is shone onto the disk • **Pits** are **burnt** into the disk • Pits are in a (spiral) track • **Pits** and lands represent binary values

2(a)	One mark for each correct nibble (2) One mark for a correct method of working for example 4 or 5 correct carries One mark for showing an overflow error (1) 111 1 11110101 + 00111001 1 0010 1110
2(b)	One mark for showing the binary number after the shift One mark for valid method of working to convert to denary One mark for correct denary number • 00011110 • 16 + 8 + 4 + 2 • 30

1(a)	1 mark for each correct item (in bold) The binary number system is base **2**. The smallest denary number that can be represented an 8-bit binary number is **0**. The largest denary number that can be represented an 8-bit binary number is **255**. The hexadecimal number system is base **16**. Each hexadecimal digit is equivalent to **4** bits. The numbers 1 to 9 are used and the number 10 is represented by **A**. The hexadecimal number system continues up to the number 15, which is represented by **F**.
1(b)	1 mark for correct working, 1 mark for each correct nibble For example: 1 0 0 1 1 0 1 1 + 0 0 0 1 0 0 1 1 1 0 1 0 1 1 1 0 1 1 1
1(c)	C

3(a)(i)	They are both **number systems**
3(a)(ii)	• Binary is base-2 whereas hexadecimal is base-16 • Binary only uses numbers whereas hexadecimal also uses letters // Binary only uses 0 and 1 whereas hexadecimal uses 0 to 9/A to F
3(b)	• (0000)1111 • 10110100 • 11101011
3(c)	• E • 64 • FA
3(d)(i)	Any **two** from: • **Each/All/Every value/digit/bit** in the binary number is **shifted**/**moved** to the **left** • The **left most**/**most significant** bit is lost • A **0** is added as the **right most/least significant** bit
3(d)(ii)	The binary integer is multiplied by 2
3(e)	Two's complement

2(a)	**C**
2(b)	14 20 A5

3(a)	00011000
3(b)	**D**
3(c)	**One** mark for correct working Example: Flip and add **One** mark for correct answer: −93
3(d)	1024

3(a)	Hexadecimal
3(b)(i)	1010 110010 11001001
3(b)(ii)	**Two** from: • Computers use **logic gates/switches** … • … that only process the values 1 and 0 // that only have two states
3(c)	**One** mark for evidence of working, for example 2 carries **One** mark for each correct nibble (Max 2) 1 1 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 0
3(d)	**One** marking for evidence of working For example, flip and add **One** mark for correct binary 11100000

1(a)	B
1(b)(i)	A
1(b)(ii)	01001110
1(b)(iii)	• **Unique** binary/denary number given/stored for each character • The code for R is stored, then the code for E then D **in sequence**
1(c)(i)	Any **two** from: • More bits allocated to each amplitude • **Amplitudes** can be more precise • A wider range of **amplitudes** can be recorded
1(c)(ii)	Increase the sample rate

8(a)(i)	EC
8(a)(ii)	Any **one** from: • Easier for humans to read/remember • Shorter for humans to enter • Less likely for humans to make mistakes • Easier for humans to spot errors/debug • Takes up less space **onscreen**
8(b)(i)	**One** mark for working, **one** mark for answer • e.g. showing flip and add 1 10110111 = 01001001 • –73
8(b)(ii)	00101101
8(c)	**One** mark each • Divide • ...by 16 / 2^4

<table>
<tr><td>2(a)</td><td>– 227</td></tr>
<tr><td>2(b)</td><td>One mark for each correct character in the correct order:
– E3</td></tr>
<tr><td>2(c)</td><td><table><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr></table></td></tr>
<tr><td>2(d)</td><td>One mark for suitable working method e.g. flip and add 1
One mark for correct answer

– 10011101</td></tr>
<tr><td>2(e)</td><td>One mark for each correct nibble (max 2)
One mark for correct working e.g. correct carry
One mark for showing overflow bit

1
1 1 1 0 0 0 1 1
+ 0 1 0 0 1 1 0 0
1 0 0 1 0 1 1 1 1</td></tr>
</table>