Data Representation

1.1 Number systems

Marking Scheme

1 **(a)** hours: **18**

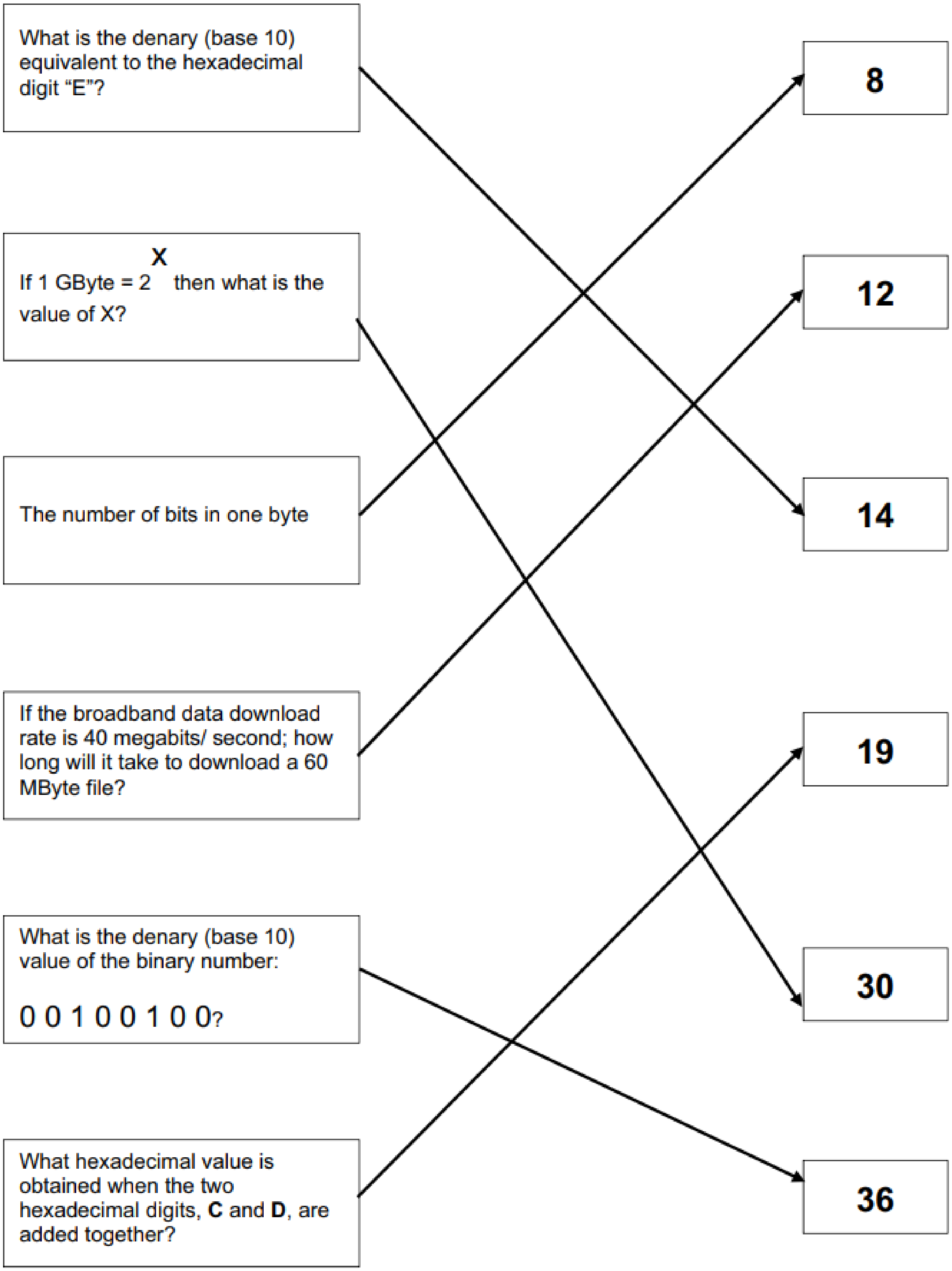
minutes: **53** [2]

(b)

hours (“**C**”)									minutes (“**D**”)							
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**:**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**

[2]

2



What is the denary (base 10) equivalent to the hexadecimal digit “E”?

If 1 GByte = 2^{X} then what is the value of X?

The number of bits in one byte

If the broadband data download rate is 40 megabits/ second; how long will it take to download a 60 MByte file?

What is the denary (base 10) value of the binary number:

0 0 1 0 0 1 0 0?

What hexadecimal value is obtained when the two hexadecimal digits, **C** and **D**, are added together?

8

12

14

19

30

36

5/6 matches – 5 marks
4 matches – 4 marks
3 matches – 3 marks
2 matches – 2 marks
1 match – 1 mark

[5]

3 **(a)** 1 mark for two correct lines, 2 marks for four correct lines

L (108):	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**
I (105):	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**
G (103):	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**
N (110):	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**

[2]

(b) 1 mark for each correct binary value
1 mark for each correct hexadecimal value

									hexidecimal
L:	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**D8**
G:	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**0**	**CE**

[4]

4 **(a)** 1 0 1 1 0 1 0 1

F 6 [2]

(b) Any **two** from:
- HTML
- MAC address
- used in assembly language/machine code
- debugging (displays bytes in hex when using memory dumps)

[2]

(c)
- Can represent 16 bit words as only 4 hexadecimal digits
- It is easy to convert hex digits back to binary if necessary

[2]

5 (a)

w	w	w	.	c	i	e	.	o	r	g	.	u	k
%77	%77	%77	**%2E**	**%63**	**%69**	**%65**	**%2E**	**%6F**	**%72**	**%67**	**%2E**	**%75**	**%6B**

1 mark 1 mark 1 mark

[3]

(b)

%77	%77	%77	%2E	%72	%6F	%63	%6B	%69	%63	%74	%2E	%63	%6F	%6D
W	W	W	.	r	o	c	k	i	c	t	.	c	o	m

1 mark 1 mark 1 mark

[3]

6 **(a)** **(i)** For each hex number, 2 marks if all correct, 1 mark for 2 correct conversions

F A 7:	**1**	**1**	**1**	**1**		**1**	**0**	**1**	**0**		**0**	**1**	**1**	**1**

D 3 E:	**1**	**1**	**0**	**1**		**0**	**0**	**1**	**1**		**1**	**1**	**1**	**0**

[4]

(ii) 2 marks if all correct, 1 mark for 2 correct conversions – Follow through

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

[2]

(iii) 2 marks if all correct, 1 mark for 2 correct conversions – Follow through
D 2 6 [2]

(b) **(i)** (X) FF FF 00

(Y) FF 00 FF

(Z) 00 FF FF [3]

(ii)
- hex values between 0 to F are combined together to create a hex code
- different combinations in hex codes will create different shades/tones/colours [2]

(c) **(i)** First six digits: manufacturer code/manufacturer ID

Last six digits: serial number/serial ID of device/product [2]

(ii) Allows all devices to be uniquely identified [1]

7 **(a)** **1** mark for each correct binary value

3	**0**	**0**	**1**	**1**
5	**0**	**1**	**0**	**1**

[2]

(b)

0	**0**	**0**	**1**	→	**1**	1 mark
1	**0**	**0**	**1**	→	**9**	
0	**1**	**0**	**0**	→	**4**	
1	**1**	**1**	**0**	→	**E**	1 mark

[2]

8 **(a)** **1** mark for each nibble

0100 1010 1111 [3]

(b)	**(i)**	0 1 1 0 1 0 0 1	105 hours	1 mark	
		0 0 0 1 1 1 1 1	31 minutes	1 mark	
		0 0 1 1 0 0 1 0	50 seconds	1 mark	[3]
	(ii)	1F			[1]

9	(a)	1 mark for **any** two correct values, 2 marks for all 4 correct values. 29FC	**2**
	(b)	**Two** from: ∞ Easier/quicker to understand/read ∞ Easier to debug/identify errors ∞ Fewer digits are used / shorter // takes up less space on screen // more can be shown on screen / page	**2**
	(c)	**Two** from: ∞ Notations for colour in HTML // HTML colour (codes) ∞ Error messages ∞ MAC address // IP address ∞ Locations in memory ∞ Memory dump	**2**

<table>
<tr><td><b>10</b></td><td>(a)</td><td>1 mark for correct method, 1 mark for correct answer

32 + 16 + 8 + 1
(00)111001</td><td><b>2</b></td></tr>
<tr><td></td><td>(b)</td><td>registers <b>must</b> have leading zeros, allow follow through from 5(a) for an incorrect value
1 mark for each correct register.

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

<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>
</td><td><b>2</b></td></tr>
<tr><td></td><td>(c)</td><td><b>Two</b> from:

∞ data
∞ ASCII value / Unicode value / character
∞ number
∞ part of image / small image
∞ a sound / sound sample / small sound track
∞ instruction</td><td><b>2</b></td></tr>
<tr><td></td><td>(d)</td><td>3A</td><td><b>1</b></td></tr>
</table>

<table>
<tr><td><b>11</b></td><td>(a)</td><td>Output</td><td><b>1</b></td></tr>
<tr><td></td><td>(b)</td><td>1 mark for each correct conversion
<table>
<tr><td colspan="4"><i>E</i></td><td colspan="4"><i>0</i></td><td colspan="4"><i>4</i></td></tr>
<tr><td><b>1</b></td><td><b>1</b></td><td><b>1</b></td><td><b>0</b></td><td><b>0</b></td><td><b>0</b></td><td><b>0</b></td><td><b>0</b></td><td><b>0</b></td><td><b>1</b></td><td><b>0</b></td><td><b>0</b></td></tr>
</table></td><td><b>3</b></td></tr>
<tr><td></td><td>(c)</td><td>Any <b>one</b> from:
– Hexadecimal codes can fit in a smaller display rather than a full text based message
– Smaller amount of memory needed to store the hex error messages than text based</td><td><b>1</b></td></tr>
<tr><td></td><td>(d)</td><td>1 mark for correct sensor, 1 mark for corresponding use
Possible examples could include:

– Temperature (sensor)
– To monitor the temperature of the water

– Pressure (sensor)
– To monitor the level of water in the washing machine

– Motion (sensor)
– To monitor whether the drum is still in motion

– pH (sensor)
– To monitor the level of water hardness/detergent present in the water</td><td><b>6</b></td></tr>
</table>

12 1 mark per correct instruction: **5**

9 – LEFT
1 – DOWN
C – OPEN
3 – CLOSE
F – UP

13 (a) Any **four** from (**Max 2** per number system) : **4**

- A binary number system is a base-2 system
- A denary number system is a base-10 system

- A binary number system uses 0 and 1 values
- A denary number system uses 0 to 9 values

- A binary number system has units/ placeholders/column headings that increase by the power of 2
- A denary number system has units/ placeholders/column headings that increase by the power of 10

- Binary has more digit <u>for the same value</u>// Denary has less digits <u>for the same value</u>

(b) **Five** from: **5**

- Correct column headings / place holders by example
- Correctly place a 1 or a 0 for each column
- Identify the columns to be added
- Add together the (denary) values identified …
- … this will give a total which is the denary number/answer
- Answer is 10

14	1 mark for each correct answer, in the given order: – analogue – digital – denary – 10 – binary – 2	**6**

15

1 mark for each correct conversion:

- 42
- 257
- 542

3

16 (a)

1 mark for each correct register

Hours	0	0	0	0	0	0	1	0
Minutes	0	0	0	1	1	1	1	1
Seconds	0	0	1	1	1	0	1	0

3

(b)

1 mark for each correct section:

0	5 :	2	6 :	5	5
Hours		Minutes		Seconds	

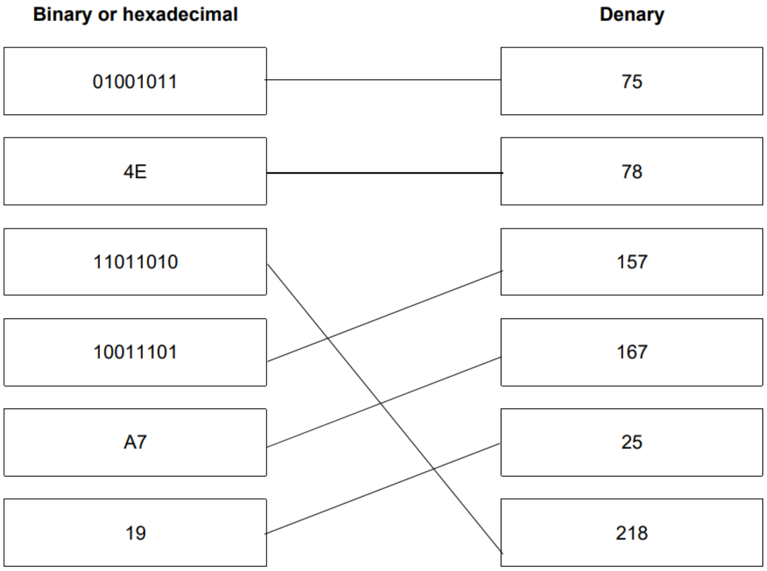
3

17 1 mark for each correct section: **3**

1	1	0	1	0	0	0	0	0	0	1	1
←— 1 mark —→				←— 1 mark —→				←— 1 mark —→			

18	(a)	(0)1101011	**1**
	(b)	000 100101100 1 mark for three leading zeros, 1 mark for correct binary number	**2**
	(c)	B3 1 mark for each correct character	**2**

19 (a) 1 mark for each correct line (to a maximum of 5) **5**



(b) **Two** from: **2**

- It makes the values easier to read/write/understand/debug
- It is a shorter way to represent the values

<table>
<tr><td><b>20</b></td><td>(a)</td><td>1 mark for each correct 8-bit binary number
<table>
<tr><td>66</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>85</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>83</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table></td><td><b>3</b></td></tr>
<tr><td></td><td>(b)(i)</td><td>1 mark for each correct hexadecimal number
4B
45
59</td><td><b>3</b></td></tr>
<tr><td></td><td>(b)(ii)</td><td><b>Three</b> from:
∞ (HTML) colour codes
∞ Error messages
∞ MAC addresses
∞ IP addresses
∞ Assembly language
∞ Memory dump
∞ Locations in memory</td><td><b>3</b></td></tr>
<tr><td></td><td>(b)(iii)</td><td><b>Two</b> from:
∞ Easier to read/write/understand (for humans)
∞ Easier to remember (for humans)
∞ Short way to represent binary // Uses less <b>screen/display</b> space
∞ Fewer errors made (in data transcription)
∞ Easier to debug (for humans)</td><td><b>2</b></td></tr>
</table>

21 (a) 1 mark for each correct conversion **3**

01101010	**11111111**	**00001000**	**10010011**

(b) **2**

∝ Computers use switches / logic gates
∞ Only uses 2 states / On or Off / 1 or 0

22 (a) **6**

97

1	0	0	1	0	1	1	1

1 mark 1 mark

5C

0	1	0	1	1	1	0	0

1 mark 1 mark

E1

1	1	1	0	0	0	0	1

1 mark 1 mark

(b) **Four** from: **4**

- Media Access Control (address)
- Used to identify a device
- It is a **unique** (address)
- It is a static address // It does not change
- It is set by the manufacturer
- The first part is the manufacturer ID/number/identifies the manufacturer
- The second part is the serial number/ID

23 (a) **6**

1	1	0	1	0	0	1	0	1	1	1	0

1 mark 1 mark 1 mark

0	0	1	1	1	1	1	1	0	1	0	1

1 mark 1 mark 1 mark

24 (a) **One** mark for each correct denary value **3**

Binary	Denary
0001001110	78
0110110111	439
1000000001	513

(b) **Two** from: **2**

- Uses fewer characters // shorter
- Easier to read / write / understand
- Less likely to make mistakes // less error prone
- Easier to debug

(c) **One** mark for each correct hexadecimal value in correct order **3**

2 B 5

25 (a) **1**

- 52

(b) **1**

1	1	0	1	0	0	0	0

(c) **1**

- It is multiplied by 4

26	(a)(i)	– 12 (ignore leading zeros)	**1**
	(a)(ii)	– 198 (ignore leading zeros)	**1**
	(a)(iii)	– 1217	**1**
	(b)	**One** mark per each correct hex value in correct order – 0E9	**3**

27	(a)	– 21 – 258 – 169	**3**
	(b)	1 mark for each correct hex value – 50 – 3D	**4**

28	(a)	**One** mark for each correct binary conversion **One** mark for each correct denary conversion	**6**

Hexadecimal ticket number	12-bit binary value	Denary value
028	**0000 0010 1000**	**40**
1A9	**0001 1010 1001**	**425**
20C	**0010 0000 1100**	**524**

29 (a)

One mark per each correct row:

Denary	Binary Conversion	Correct (✓)	Incorrect (✓)
145	10010001	✓	
179	10110101		✓
11	00010011		✓
100	01100010		✓

4

(b)

One mark for each correct conversion in the correct order**:**

- C
- 4
- 0

3

30 (a)

One mark per each correct binary value.
One mark per each correct hex value.

Denary	Hexadecimal	8-bit binary
49	31	00110001
123	7B	01111011
200	C8	11001000

6

(b)

Any **two** from:

- Easier/quicker to read/write/understand
- Easier/quicker to identify errors/debug
- Takes up less **screen/display** space
- Less chance of making an error

2

(c)

Any **three** from:

- MAC address
- URL
- Assembly language
- Error codes // error messages
- IP addresses
- Locations in memory
- Memory dumps

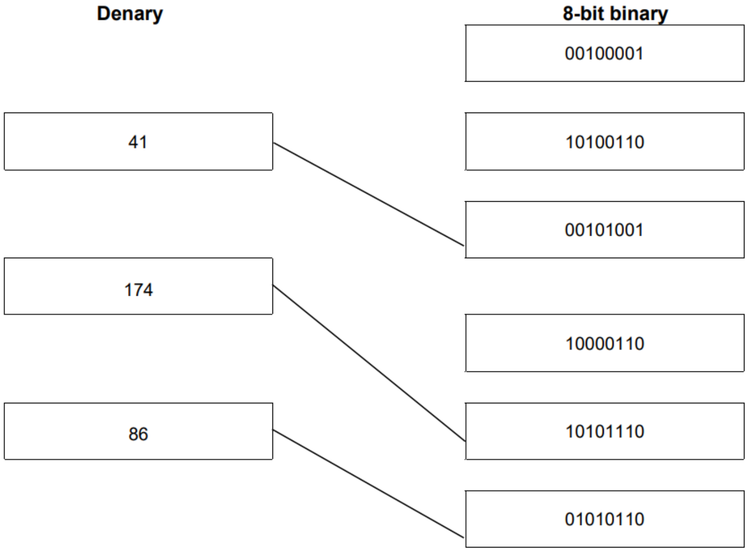
3

31	(a)	– Base-2	**1**
	(b)	– 9 – 16 – 40 – 161	**4**

32	(a)	– Base-10	**1**
	(b)	– 5 – 32 – 26 – 171	**4**
	(c)(i)	– 00100101	**1**
	(c)(ii)	– 00011011	**1**
	(d)(i)	Any **one** from: – To represent **HTML colour** codes – In error messages	**1**
	(d)(ii)	Any **one** from: – Assembly code/language – Memory address locations – In error messages – Memory dump	**1**

<table>
<tr><td>33</td><td>(a)</td><td><b>One</b> mark for correct binary value, <b>one</b> mark for leading zeros
00000000 01000111</td><td><b>2</b></td></tr>
<tr><td></td><td>(b)</td><td><b>One</b> mark for leading zeros, <b>one</b> mark for correct binary value
00000001 00000001</td><td><b>2</b></td></tr>
<tr><td></td><td>(c)</td><td>– 0516</td><td><b>1</b></td></tr>
<tr><td></td><td>(d)(i)</td><td>– Pressure sensor
– Motion sensor</td><td><b>2</b></td></tr>
<tr><td></td><td>(d)(ii)</td><td><b>One</b> mark for the correct tick
<table>
<tr><th>Device</th><th>Tick (✓)</th></tr>
<tr><td>input</td><td>✓</td></tr>
<tr><td>storage</td><td></td></tr>
<tr><td>output</td><td></td></tr>
</table></td><td><b>1</b></td></tr>
</table>

34	(a)	• Computer consist of transistors / logic circuits/gates … • … that can only **store/process** data in **two states / high-low / on-off / 1 and 0**	**2**
	(b)	• 01000000 • 01100101 • 11110010	**3**
	'c)	• 0100 (1 mark) 0010 (1 mark) • 1100 (1 mark) 1110 (1 mark)	**4**

35	(a)	**One** mark for each correct line  Denary 8-bit binary 41 174 86 00100001 10100110 00101001 10000110 10101110 01010110	**3**
	(b)	**One** mark for correct working, **one** mark for correct answer Working e.g. • 256 + 64 + 16 + 4 + 2 + 1 Answer: • 343	**2**

36 (a) **Two** marks each correct conversion (**one** mark for the first four bits, **one** mark for the second four bits) **6**

2F	0	0	1	0	1	1	1	1
15	0	0	0	1	0	1	0	1
D6	1	1	0	1	0	1	1	0

(b) Any **two** from: **2**

- IP address
- Error messages/codes
- Assembly language // low-level language
- URL // web address
- Memory dumps
- Locations in memory

(c) **One** mark for a description, **one** mark for a correct example **4**

Structure

- **Layout** of the web page
- e.g. Where text is placed

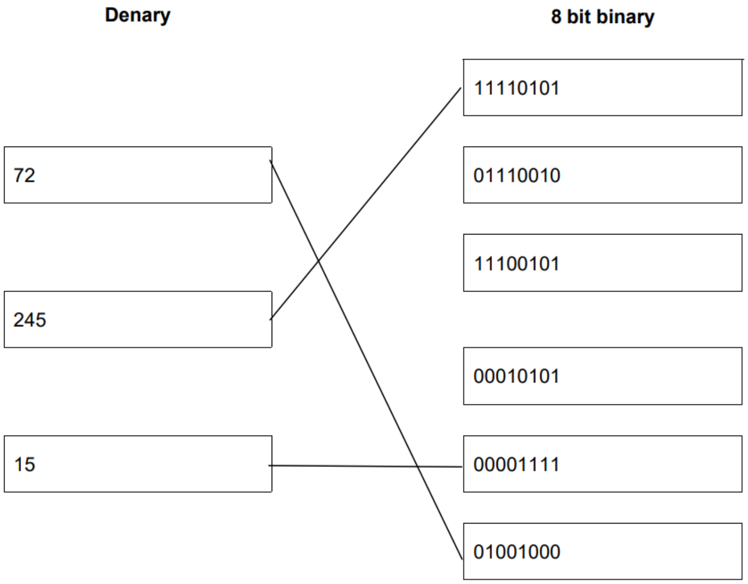
Presentation

- **Formatting** of the web page
- e.g. the colour of the font

37	(a)	**One** mark for each correct bus (max 2) and **one** mark for corresponding description of transmission • Data bus • …. responsible for transmitting data/instructions • Control bus • … responsible for transmitting control signals	**4**
	(b)	Any **one** from: • Fetch • Decode	**1**
	(c)	Any **two** from: • To **temporarily** store data • It stores the **result** of interim **calculations** **One** from: • Arithmetic logic unit / ALU	**3**

38	(a)	**One** mark for two correct characters, **two** marks for three, in the correct place • 0100 0000 0100	**2**
	(b)	**One** mark for two correct characters, **two** marks for three • 0001 0010 1011	**2**
	(c)	**One** mark for each correct denary conversion • 34 • 172	**2**
	(d)	**One** mark for two correct characters, **two** marks for three, in the correct place • 9E0	**2**
	(e)	Any **two** from: • It is easier for user to read/recognise/understand • It takes up less space on a **display**	**2**

39	(f)(i)	• 000001100100 • 000011101011 • 000100101101	**3**
	(f)(ii)	• 22 • 119 • 857	**3**
	(f)(iii)	**One** mark for two correct characters in the correct place, **two** marks for three • 095 • AD1	**4**

40	(a)	**One** mark for each correct line  Denary 8 bit binary 72 245 15 11110101 01110010 11100101 00010101 00001111 01001000	**3**
	(b)	**One** mark for two correct characters, **two** marks for three correct characters, **three** marks for four correct characters, in the correct place • 09AE	**3**

41

Question	Answer	Marks
(a)	• B	**1**
(b)	**One** mark per each correct conversion • 00110010 • 01100110 • 11011101	**3**
(c)	**One** mark for full method of working e.g. conversion to binary then flipping and adding 1 **One** mark for correct answer • 10110010	**2**
(d)	**One** marks per each correct nibble **One** mark for correct working in binary (showing 4 correct carries) 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 0 1 10 0 1 0 1 0 0	**3**
(e)	**Two** from: • The result of the calculation is greater than 255 // The value generated is larger than can be stored in the register • The result of the calculation would require more than **8 bits** to be represented // A **register** has a predetermined **number of bits** and there are **too many bits** for it	**2**

42

Question	Answer	Marks
(a)	**One** mark per each correct character in the correct order: • 9 • 3 • 0 • D	**4**
(b)(i)	• 00001111	**1**
(b)(ii)	Any **one** from: • The value becomes incorrect/inaccurate as the right most bits are lost • It is divided by 8	**1**
(c)	Any **two** from: • Easier/quicker to understand/read/write • Easier/quicker to debug • Less likely to make a mistake • Shorter representation // Takes up less **screen** space	**2**
(d)	**One** mark for two correct characters, **two** marks for three correct characters in the correct order: • 1 • 2 • D	**2**

43

Question	Answer	Marks
(a)	• 174	**1**
(b)	• A • E	**2**
(c)(i)	• 01110000	**1**
(c)(ii)	• B	**1**
(d)	**One** mark for each correct nibble **One** mark for correct carries (or other correct working method) **One** mark for identification of overflow error 1 1 • 1 0001 1111	**4**
(e)	• 9	**1**
(f)	• 12	**1**

44

Question	Answer	Marks
(a)	**One** mark for each correct part of the fee, in the correct order: – 17 – 70 (Correct fee $17.70)	**2**
(b)	**One** mark for each correct binary value: Register 1 – 00001110 Register 2 – 01100010	**2**

Question	Answer	Marks
(c)	**One** mark for each correct hexadecimal value, in the correct order. – A – 0 – 3 – D (Ticket number A03D)	**4**
(d)	**Two** from: – It contains **logic gates/switches** … – … that process the values 1 and 0 // have two states	**2**
(e)	Any **four** from: – Compares the ticket number received to stored data – … that is a **database/file** of ticket numbers – … checks the ticket number is listed as paid – If the **data** matches/cost is paid it sends a **signal** to raise the barrier – If the **data** does not match/cost is not paid, the barrier remains down	**4**

45

Question	Answer	Marks
(a)	– 227	**1**
(b)	**One** mark for each correct character in the correct order: – E3	**2**
(c)	1 | 0 | 0 | 0 | 1 | 1 | 0 | 0	**1**
(d)	**One** mark for suitable working method e.g. flip and add 1 **One** mark for correct answer – 10011101	**2**
(e)	**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	**4**

46

Question	Answer	Marks
(a)	Any **two** from: – It has a base of 2 – It only uses two **values** – … that are 1 and 0	**2**
(b)	– (0000)1110 – (00)111011 – 11101010	**3**
(c)	– 9 – 1A – 41	**3**
(d)	**One** mark for suitable working method e.g. conversion to binary **One** mark for correct answer – 01111011	**2**
(e)	**One** mark for each correct nibble (max 2) **One** mark for correct working e.g. correct carries 1 1 1 0 0 1 1 0 0 1 1 + 0 1 1 1 1 0 0 0 1 0 1 0 1 0 1 1	**3**

47

Question	Answer	Marks
(a)	**C**	**1**
(b)	14 20 A5	**3**

48

Question	Answer	Marks
(a)	00011000	**1**
(b)	**D**	**1**
(c)	**One** mark for correct working Example: Flip and add **One** mark for correct answer: −93	**2**
(d)	1024	**1**

49

Question	Answer	Marks
(a)	Hexadecimal	**1**
(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	**2**
(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	**2**

50

Question	Answer	Marks
(a)(i)	**One** mark for each correct nibble, in the correct order. 1010 0010 1111	**3**
(a)(ii)	2607	**1**
(b)(i)	**One** mark for each correct character, in the correct order. 1 9 B	**3**
(b)(ii)	411	**1**
(c)	Any **one** from: • It is easier/quicker to read/understand/debug • It is a shorter representation of binary // It takes up less **screen** space	**1**
(d)	Any **two** from: Example: • **HTML** colour codes • URL • Memory dump • IP address • MAC address • Assembly language • Error codes/messages • ASCII/Unicode	**2**
(e)	**One** mark for correct working: Example: flip and add **One** mark for correct answer. 11100111	**2**

51

Question	Answer	Marks
(a)	Unicode	**1**
(b)(i)	• (0)1000001 • (0)1101101	**2**
(b)(ii)	• 41 • 6D	**2**
(c)(i)	121	**1**
(c)(ii)	79	**1**
(c)(iii)	00011110	**1**
(d)	**One** mark for correct working, for example: carries **One** mark for each correct nibble. 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0 0 0	**3**

52

Question	Answer	Marks
(a)(i)	They are both **number systems**	**1**
`(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	**2**

Question	Answer	Marks
(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	**2**
(d)(ii)	The binary integer is multiplied by 2	**1**
(e)	Two's complement	**1**

53	(c)(i)	**One** mark for valid working, for example: 128 + 32 + 8 + 4 + 2 + 1 **One** mark for correct answer: 10101111	**2**
	(c)(ii)	• 0001 0101 • 0010 1101 • 0000 1001 0001	**3**
	(d)	**One** mark for each correct nibble. **One** mark for method of working, for example: carries. **One** mark for identification of overflow. 1 1 1 1 0 0 0 1 1 + 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 1	**4**
	(e)	**One** mark for correct working, for example: flip and add **One** mark for correct denary. –114	**2**