2.9 Data Representation and Theory of Computation: 30-minute Pack

Total Marks

AQA 7517 · Calculators not allowed · about 30 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 A microcontroller stores values in an 8-bit unsigned binary register, so it can only hold values from 0 to 255.

(a) Add the two unsigned 8-bit binary numbers 01101011 and 00111001, showing your working column by column (including any carries), and give the 8-bit result. **(3)**

(b) State, with a reason, whether an overflow has occurred in part (a). **(2)**

(c) Add the two unsigned 8-bit binary numbers 11001101 and 01101011, showing your working, and state whether overflow occurs this time, explaining your reasoning. **(3)**

(Total for Question 1 is 8 marks)

---

2 An embedded sensor stores fractional readings using an 8-bit fixed-point binary format, with 4 bits for the integer part and 4 bits for the fractional part (format IIII.FFFF, unsigned).

(a) Convert the fixed-point binary number 0101.1100 to denary, showing your method. **(2)**

(b) Convert the denary value 9.375 to this 8-bit fixed-point binary format (IIII.FFFF), showing your method for both the integer and fractional parts. **(3)**

(c) State the largest positive denary value that can be represented in this 8-bit format, showing your working. Then explain why increasing the number of fractional bits (keeping the total word length at 8 bits) would improve precision but reduce the range of values that can be represented. **(3)**

(Total for Question 2 is 8 marks)

© Revision Library | revisionlibrary.co.uk

3 A text file's character encoding is being reviewed for compatibility with different languages.

(a) State the number of bits used to represent one character in 7-bit ASCII, and calculate how many different characters can be represented in this scheme. **(2)**

(b) A student states that 7-bit ASCII cannot represent the character e with a circumflex (as used in French), or emoji such as a smiley face. Explain why Unicode was developed to address this limitation, and state one advantage Unicode has over extended (8-bit) ASCII. **(3)**

(c) A document contains 3,200 characters. Calculate the file size, in bytes, needed to store this document using: (i) 7-bit ASCII, with each character stored in a full byte, and (ii) UTF-32 encoding, with each character stored using 32 bits. **(3)**

(Total for Question 3 is 8 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.9 Data Representation and Theory of Computation: 30-minute Pack

Question 1

(a) binary addition attempted with carries shown for each column **M1**

(a) carries correct throughout (carry sequence, from bit0: 1,1,0,1,1,1,1, final carry out 0) **A1**

(a) 10100100 cao **A1**

(a) Answer: 10100100

(b) no overflow has occurred **B1**

(b) correct reason, e.g. there is no carry out of the most significant (bit 7) column, or the true sum 164 fits within the unsigned 8-bit range 0 to 255 (oe) **B1**

(b) Answer: No overflow. The final carry out of bit 7 is 0, and the true sum, 164, fits within the unsigned 8-bit range of 0 to 255.

(c) binary addition attempted with carries shown for each column **M1**

(c) 00111000 correctly given as the 8-bit result (i.e. the final carry out of bit 7 is discarded) **A1**

(c) overflow correctly stated to occur, dependent on a correct or near-correct addition, with correct reason: there is a carry out of the most significant (bit 7) column (a 9th bit is needed), because the true sum, 312, exceeds the maximum unsigned 8-bit value of 255 (oe) **dM1**

(c) Answer: Result (8-bit): 00111000. Overflow occurs, because the true sum is 205+107=312, which exceeds 255 (the largest value 8 bits can hold), so a carry is generated out of bit 7 that the register cannot store.

Question 2

(a) integer part correctly identified as 5, and fractional part correctly weighted (1/2 + 1/4 = 0.75) **M1**

(a) 5.75 cao **A1**

(a) Answer: 5.75

(b) integer part correctly converted: 9 = 1001 **B1**

(b) fractional part correctly converted using repeated doubling: 0.375 = 0110 (0.375x2=0.75 -> 0, 0.75x2=1.5 -> 1, 0.5x2=1.0 -> 1, 0x2=0 -> 0) **B1**

(b) 1001.0110 correctly combined cao **B1**

(b) Answer: 1001.0110

(c) correct largest value 15.9375, with working shown (integer part 1111=15, fractional part 1111=15/16=0.9375) **B1**

(c) precision correctly explained: more fractional bits give a smaller gap between adjacent representable values, i.e. finer resolution (oe) **B1**

(c) range correctly explained: with a fixed 8-bit word length, giving more bits to the fractional part leaves fewer bits for the integer part, reducing the largest (and smallest) value that can be represented (oe) **B1**

(c) Answer: Largest value: 15.9375 (integer part 1111=15, fractional part 1111=15/16=0.9375). More fractional bits improve precision (smaller gaps between representable values) but, because the total word length is fixed, this takes bits away from the integer part, reducing the overall range.

Question 3

(a) 7 bits stated **B1**

(a) 128 correctly calculated (2^7) cao **B1**

(a) Answer: 7 bits; 2^7 = 128 characters.

(b) explains why ASCII is limited: 7-bit ASCII only encodes 128 characters, mostly basic Latin letters, digits and control characters, so it has no code points for accented letters, other scripts (e.g. Chinese, Arabic, Cyrillic) or symbols such as emoji (oe) **B1**

(b) describes what Unicode does: it gives every character, in every writing system (and many symbols), a unique code point, so text in any language or script can be represented consistently (oe) **B1**

(b) valid stated advantage over extended ASCII, e.g. Unicode supports a vastly larger number of characters (extended ASCII is limited to 256), allowing many different languages/scripts to be mixed reliably in the same document (oe) **B1**

(b) Answer: 7-bit ASCII only has 128 code points, covering basic Latin characters, so it cannot represent accented letters, other scripts, or symbols like emoji. Unicode gives every character in every script (and many symbols) a unique code point. Advantage: Unicode supports far more than the 256 characters of extended ASCII, so many languages can be represented consistently in one standard.

(c) correct method: number of characters x bytes per character, for each scheme (1 byte for (i), 4 bytes for (ii)) **M1**

(c) (i) 3200 bytes cao **A1**

(c) (ii) 12800 bytes cao **A1**

(c) Answer: (i) 3200 x 1 = 3,200 bytes. (ii) 3200 x 4 = 12,800 bytes.

© Revision Library | revisionlibrary.co.uk