2.9 Trace Tables and Binary/Hex Conversions: 30-minute Pack

Total Marks

OCR J277 · Calculators not allowed · about 30 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Convert each of the following 8 bit binary numbers to a two digit hexadecimal number.

(a) Convert 10110101 to hexadecimal. Show your working. **(2)**

(b) Convert 01111110 to hexadecimal. Show your working. **(2)**

(Total for Question 1 is 4 marks)

---

2 Programmers often write memory addresses and colour codes in hexadecimal rather than binary.

(a) State two reasons why hexadecimal is often used instead of binary, and give one real-world example of where hexadecimal is used. **(3)**

(Total for Question 2 is 3 marks)

© Revision Library | revisionlibrary.co.uk

3 The 8 bit binary number 00001101 is shifted left by 2 places.

(a) Calculate the 8 bit binary result of the shift. **(2)**

.....

(b) State the denary value before and after the shift, and hence state the general effect of a left shift of n places on the value of a binary number. **(2)**

(Total for Question 3 is 4 marks)

---

4 Some software needs to store characters from many different alphabets, including Mandarin and Arabic characters, as well as emoji.

(a) Explain why Unicode, rather than standard 7 bit ASCII, is used to represent these characters. **(2)**

.....

.....

(b) 16 bit Unicode is used instead of standard 7 bit ASCII. Calculate how many more bits per character 16 bit Unicode uses compared to 7 bit ASCII. **(2)**

.....

(Total for Question 4 is 4 marks)

© Revision Library | revisionlibrary.co.uk

5 The algorithm below uses OCR reference language pseudocode:

```
total <- 0
FOR count <- 1 TO 5
    total <- total + (count * 2)
NEXT count
OUTPUT total
```

(a) Complete the trace table below to show the value of total after each pass of the loop, and state the value that is output. **(6)**

count	total
(start)	0
1	
2	
3	
4	
5	

```
OUTPUT =
```

(Total for Question 5 is 6 marks)

© Revision Library | revisionlibrary.co.uk

6 The algorithm below uses OCR reference language pseudocode to convert a denary number to binary:

```
number <- 44
binaryDigits <- ""
WHILE number > 0
    remainder <- number MOD 2
    binaryDigits <- STR(remainder) + binaryDigits
    number <- number DIV 2
ENDWHILE
OUTPUT binaryDigits
```

(a) Complete the trace table below to show the values of remainder, binaryDigits and number after each pass of the loop, and state the string that is output. Then verify your answer by converting it back to denary. **(9)**

remainder	binaryDigits	number
-	""	44 (start)

```
OUTPUT =
```

(Total for Question 6 is 9 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.9 Trace Tables and Binary/Hex Conversions: 30-minute Pack

Question 1

(a) Binary split into two correct nibbles: 1011 and 0101, oe **M1**

(a) B5, cao **A1**

(a) Answer: B5

(b) Binary split into two correct nibbles: 0111 and 1110, oe **M1**

(b) 7E, cao **A1**

(b) Answer: 7E

Question 2

(a) hexadecimal is much shorter/more compact to write than binary for the same value, oe **B1**

(a) each hex digit represents exactly one nibble (4 bits), so conversion between binary and hex is quick and direct, and hex is easier for humans to read/less error-prone to transcribe than long strings of 0s and 1s, oe **B1**

(a) valid real-world example, e.g. MAC addresses, HTML colour codes, memory addresses/locations (any suitable example) **B1**

(a) Answer: Hex is shorter and easier to read than binary, and each digit maps directly to 4 bits, e.g. it is used for MAC addresses or HTML colour codes.

Question 3

(a) correct shifting process shown, e.g. bits moved two places left, two 0s inserted on the right, oe **M1**

(a) 00110100, cao **A1**

(a) Answer: 00110100

(b) before = 13, after = 52, both correct, oe **B1**

(b) a left shift by n places multiplies the value by 2^n (here, shifting left by 2 multiplies by 4), provided no significant bits are lost off the left hand end, oe **B1**

(b) Answer: Before = 13, after = 52; a left shift by n places multiplies the value by 2^n.

Question 4

(a) standard ASCII only has enough bits to represent a limited number of characters (128 with 7 bits), which is not enough to cover every character used in every language and symbol set, oe **B1**

(a) Unicode uses more bits per character, so it can represent a much larger number of unique characters, covering characters from many different languages and other symbols such as emoji, oe **B1**

(a) Answer: ASCII cannot represent enough characters for every language; Unicode uses more bits per character, allowing far more characters to be represented.

(b) 16 - 7 shown as the method, oe **M1**

(b) 9, cao **A1**

(b) Answer: 9 more bits per character

Question 5

(a) count = 1: total = 2, cao **B1**

(a) count = 2: total = 6, cao **B1**

(a) count = 3: total = 12, cao **B1**

(a) count = 4: total = 20, cao **B1**

(a) count = 5: total = 30, cao **B1**

(a) OUTPUT = 30, cao **B1**

(a) Answer: total: 2, 6, 12, 20, 30; OUTPUT = 30

Question 6

(a) iteration 1: remainder = 0, binaryDigits = "0", number = 22, cao **B1**

(a) iteration 2: remainder = 0, binaryDigits = "00", number = 11, cao **B1**

(a) iteration 3: remainder = 1, binaryDigits = "100", number = 5, cao **B1**

(a) iteration 4: remainder = 1, binaryDigits = "1100", number = 2, cao **B1**

(a) iteration 5: remainder = 0, binaryDigits = "01100", number = 1, cao **B1**

(a) iteration 6: remainder = 1, binaryDigits = "101100", number = 0, cao **B1**

(a) loop correctly identified as ending here, because number is no longer greater than 0, oe **B1**

(a) OUTPUT = 101100, cao **B1**

(a) correct verification: 101100 = 32+8+4 = 44, matching the original value of number, oe **B1**

(a) Answer: binaryDigits after each pass: 0, 00, 100, 1100, 01100, 101100; OUTPUT = 101100

© Revision Library | revisionlibrary.co.uk