2.5 Data Representation: Number Systems: 30-minute Pack

Total Marks

OCR J277 · Calculators not allowed · about 30 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Because 4 bits (a nibble) can represent exactly 16 different values (0-15), each hexadecimal digit maps directly onto one 4-bit nibble.

(a) Convert the 8-bit binary number 10110101 to hexadecimal by splitting it into nibbles. Show your working. **(2)**

(b) Convert the hexadecimal number E3 to an 8-bit binary number. Show your working. **(2)**

(Total for Question 1 is 4 marks)

2 Programmers often display memory addresses, error codes and colour values in hexadecimal rather than in binary or denary. Explain why programmers often use hexadecimal rather than binary or denary.

(Total for Question 2 is 3 marks)

© Revision Library | revisionlibrary.co.uk

3 Units of data storage are built up from the bit: 1 nibble = 4 bits, 1 byte = 8 bits, and larger units (kilobyte, megabyte, gigabyte) are each 1000 times the previous unit.

(a) Put these units in order of size, smallest first: byte, bit, kilobyte, nibble. **(1)**

(b) State the number of bits in one byte. **(1)**

(c) A downloaded document is 4 kilobytes (KB) in size. Using 1 KB = 1000 bytes, calculate the number of bits in the document. Show your working. **(2)**

.....

(Total for Question 3 is 4 marks)

---

4 Show that the 8-bit binary number 11001001 represents the denary value 201.

(Total for Question 4 is 2 marks)

© Revision Library | revisionlibrary.co.uk

5 The table below shows the same three numbers written in denary, 8-bit binary and hexadecimal. Some values are missing.

Denary	Binary (8-bit)	Hexadecimal
100	?	?
?	01011001	?
?	?	7D

Complete the table by filling in every missing value. Show your working for at least one conversion in each row.

(Total for Question 5 is 6 marks)

© Revision Library | revisionlibrary.co.uk

6 A programmer wants to write an algorithm, using the OCR Exam Reference Language, that inputs two positive integers, num1 and num2, calculates their total, and outputs whether adding these two numbers together would cause an overflow if the result were stored in an 8-bit register (i.e. whether the total is greater than 255). Write this algorithm. Any correct logic will be credited; minor syntax errors will be ignored.

(Total for Question 6 is 4 marks)

7 Discuss the benefits of using hexadecimal, rather than binary or denary, when programmers need to work with large binary values such as memory addresses, error codes or colour values. In your answer, you should refer to the readability of the numbers, the length of the numbers, and how easily hexadecimal can be converted to and from binary.

(Total for Question 7 is 6 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.5 Data Representation: Number Systems: 30-minute Pack

Question 1

(a) binary correctly split into nibbles 1011 and 0101 **M1**

(a) B5 cao **A1**

(a) Answer: B5

(b) E = 1110 and 3 = 0011 identified **M1**

(b) 11100011 cao **A1**

(b) Answer: 11100011

Question 2

hexadecimal numbers are much shorter/more compact than the equivalent binary number oe **B1**

long strings of binary digits are difficult for people to read/write accurately, so hexadecimal reduces the chance of transcription errors oe **B1**

each hexadecimal digit maps directly onto a 4-bit nibble, so converting between binary and hexadecimal is quick and does not need a calculator oe **B1**

Answer: Hexadecimal is much shorter than binary for the same value, so it is quicker and less error-prone for people to read and write. It also converts to and from binary very quickly, because each hex digit represents exactly one 4-bit nibble.

Question 3

(a) bit, nibble, byte, kilobyte cao (all four correct, in this order) **B1**

(a) Answer: bit, nibble, byte, kilobyte

(b) 8 cao **B1**

(b) Answer: 8

(c) 4 x 1000 x 8 shown, or equivalent staged working **M1**

(c) 32000 cao **A1**

(c) Answer: 32000 bits

Question 4

correct place values identified: 128, 64, 8 and 1 **M1**

128+64+8+1 = 201 shown cso **A1**

Answer: 201 (given, cso)

Question 5

row 1 binary: 01100100 cao **B1**

row 1 hexadecimal: 64 cao (ft from their binary if consistent) **B1**

row 2 denary: 89 cao **B1**

row 2 hexadecimal: 59 cao (ft from the given binary) **B1**

row 3 denary: 125 cao **B1**

row 3 binary: 01111101 cao (ft from their denary if consistent) **B1**

Answer: Row 1: 100 = 01100100 = 64 (hex); Row 2: 01011001 = 89 = 59 (hex); Row 3: 125 = 01111101 = 7D (hex)

Question 6

inputs num1 and num2 and converts them to integers, e.g. using input() and int() oe **B1**

calculates the total, e.g. total = num1 + num2 oe **B1**

uses selection to compare total against 255, e.g. if total > 255 then ... else ... endif oe **B1**

outputs an appropriate message for both the overflow branch and the no-overflow branch oe **B1**

Answer:num1 = input("Enter the first number: ")
num2 = input("Enter the second number: ")
num1 = int(num1)
num2 = int(num2)
total = num1 + num2
if total > 255 then

```
    print("Overflow: the result cannot be stored in a
```

else

```
    print("No overflow. Result = " + str(total))
```

endif

© Revision Library | revisionlibrary.co.uk

Question 7

Level 1 Basic, isolated point(s) with little or no development, e.g. a simple unsupported statement that hexadecimal is shorter than binary. Answer may be a simple, undeveloped list. **1-2**

Level 2 A reasonable discussion covering at least two relevant points (e.g. length, readability, conversion), with some development but limited linkage between ideas or minor inaccuracies. **3-4**

Level 3 A detailed, coherent discussion covering multiple relevant points, clearly linked to why they matter for programmers (e.g. reduced transcription errors, quick conversion via nibbles), with logically structured reasoning throughout. **5-6**

Indicative content:

- Hexadecimal numbers are much shorter/more compact than the equivalent binary number (e.g. an 8-bit binary number needs only 2 hex digits), making them quicker to read and write.
- Long strings of binary digits are difficult for humans to read accurately and are more likely to lead to transcription/typing errors; hexadecimal reduces this risk.
- Each hexadecimal digit corresponds exactly to a 4-bit nibble, so conversion between binary and hexadecimal is quick and does not require a calculator or long division.
- Hexadecimal is still closely related to the underlying binary representation (unlike denary), so it remains meaningful to programmers working with bit patterns, memory addresses and colour codes.
- Converting between denary and binary requires more complex division/remainder calculations and is less directly linked to the bit pattern actually stored by the computer.
- A balanced answer might note that hexadecimal is still less immediately meaningful to non-programmers than denary, so it is a trade-off between compactness/accuracy for programmers and general readability.

© Revision Library | revisionlibrary.co.uk