5.4 Binary number system part 2 Mark Scheme

Q1.

(a) (i) 52; **1**

(b) (i) ‘4’ // 4 ; **1**

(ii) UNICODE // EBCDIC // EBCD // extended binary coded decimal // extended binary coded decimal interchange code;
A minor misspelling of EBCDIC **1**

(c) (i) Each pixel stored in several bits/one byte/one word;
Each colour represented by a different value; **2**

(ii) Endpoints // a pair of / two (x,y) co-ordinates // start point, direction and length;
Type of object / shape;
Thickness of shape / line;
Colour of shape/line;
A Properties of shape/line on its own; **3**

[8]

Q2.

(a) BE4;

Must be capital letters **1**

(b) 190.25 / 190 ¼ ;;;

1 mark for correct integer part,
1 mark for correct fractional part
1 mark for correct working

(e.g. correct place values) **3**

(c) –1052;;

1 mark for workings if result incorrect
1 mark for sign, 1 mark for 1052 **2**

(d) (i) -8.25 / -8¼;;;

Partial marks for workings if result incorrect
1 mark for sign, 1 mark for moving binary point 4 places or showing 2^4 **3**

(ii) Starts with 1 0

The first 2 binary digits are different;
A significant bit is stored after the (implied) binary point;
Bit after (implied) binary point different from bit before binary point;
A all leading 1's have been removed // there are no leading 1's;
R there are no leading zeros

1

[10]

Q3.

(a)

18: 0000 0000 0001 0010;

–6 1111 1111 1111 1010;

----

12 0000 0000 0000 1100; *if previous binary patterns correct*

1 mark for showing 16 bits throughout

4

(b) (i)

Symbolic Address	Hexadecimal Representation	Binary Representation	Decimal Value
Num2	A802	**1010 1000 0000 0010;**	**-2.75;;;**

If answer wrong give:
Moving e places to right / exponent processing 2^e or equivalent: *1 mark*
Correctly identifying negative number
1 mark
Follow through if binary representation wrong

4

(ii) To maximise precision in a given number of bits // to minimise rounding errors // to have just one representation of the decimal number // to simplify arithmetic operations;
A to maximise accuracy in a given number of bits;

1

[9]

Q4.

(a) 111111;

1

(b) $256/2^8$;

1

(c) $255/2^8$-1/ 11111111;

1

[3]

Q5.

(a) B76;

R lower case B

1

(b) $183^{3}/_{8}$;;
183.375;;

1 mark for correct integer part,
1 mark for correct fractional part

2

(c) (i) -36.5;;;

Partial marks for workings if result incorrect:
1 mark for x2^{6}; accept showing that binary point moves 6 places right;
1 mark for negative number;

3

(ii) A significant bit is stored after the binary point; bit after point different to bit before point; negative number starts with 10… positive number starts with 01….; to max

1

(iii) To maximise accuracy / number stored with maximum precision;
A more accurate;
A given number can only be expressed in one way in a given number of bits;

1

[8]

Q6.

(a) 1024 / 2^{10};
A 10000000000_2 *(10 0's)*

1

(b) (i) 1111111111111111; *(16 1's)*
A FFFF;
A 65,53<u>5</u> / 2^{16}<u>-1</u>;

1

(ii) 0000 0000 0010 0101; *accept if leading zeros not given*

1

(c) (i) 0011 0011 1011 0111;;; *accept 37 transposed*: 1011 0111 0011 0011;;;

1 mark for parity bits - one mark for each correct character code

f.t. for parity bits: if even number of 1's in each byte ;

3

(ii) Parity bit is set when character first generated;
(Parity bit is adjusted to make) number of 1's /on-bits even;
Parity bit is regenerated / the number of 1's is checked by receiver;
If parity bit does not match / if there are an odd number of 1's an error has occurred;

2

[8]

Q7.

(a) (i) Positive 1

(ii) $<2^{-2}$ 1

(b) Correct answer 194.5
OR 194 ½ (2)
Working (1)

If wrong answer, method marks as follows:
exponent 2^8 clearly identified (1)
*application of shift / *2^8 from correct start point (1)*
correct interpretation of bits (1)

Basically here, if it is a little inaccurate, give 2 marks, if quite inaccurate but slightly correct give 1. 3

(c) (i) Processing fixed point numbers is quicker than floating point / less processing required;
More accurate/greater precision;(1)

(ii) Where the possible range of numbers to be stored is limited / small;
Where number is of a set format / processing integers / Working with currency;
Where maximum precision is required(1) 2

[7]

Q8.

(a) The set / list of bit patterns / binary codes representing machine operations;
The set / list of bit patterns / binary codes for which machine operations have been defined;
The collection of different operations available;
A commands
R interpreted
R A set / collection etc 1

(b) 64 or 2^6 1

[2]

Q9.

(a) (i) 23; 1

(b) (i) 1010 0001;;

1 mark for correct ASCII code, one mark for odd parity bit (follow through) 2

(ii) 11010 00010 *OR* 01010 00011 *OR*

Allow stop bit to be 1 or 0 but stop and start bits must be different
Follow through if (i) wrong

01000 01011 *OR* 11000 01010;

Allow both ways round for transmission

1

[4]

Q10.

(a) (i) 1011 1101 1001 0011;

1

(ii) 1011101000 000011

-ve number; (1)

exponent +3; (*explained or demonstrated*) (1)

value 4 3/8; (1)

Answer –4 3/8 / -4.375

1 mark for each of three points

Max 3

(b) Normalisation ensures the maximum possible accuracy for a given number of bits; (*given no. of bits can be implied – e.g. leading zeroes can be replaced by significant digits at the end of the mantissa*)

Arithmetic operations simplified
Ensures that only a single representation of a number is possible;

Max 2

[6]

Q11.

(a) (i) 54;

1

(b) (i) ‘4’ / 4 ; ;

1 mark for ASCII value 52; 2 marks for correct character 4 ; ;

Max 2

(ii) UNICODE / EBCDIC / EBCD /extended binary coded decimal ;
A minor misspelling of EBCDIC

1

(c) (i) Bit-mapped graphic;
R as pixels
R jpeg *etc*

1

(ii) Image broken down into separate pixels;
Each pixel is either black or white / on or off;
Use 2 different values for black and white / 1 for black and 0 for white (or vice versa);

Store in one bit / bits / byte of computer memory;
A diagram which maps onto above points
A follow through from (i) a .gif or .jpeg image:

Max 2

[7]

Q12.

(a) (i) Any whole number. There should be no decimal point.

1

(ii) Any number with a decimal point

1

(iii) 1101.11 = 8 + 4 + 1 + ½ + ¼

=13.75 (13 ¾)

1 mark for complete working, 1 mark for answer

2

(b) B7 3E

1

(c) To represent the address / contents of a location;
Error messages;
In assembly language programs;
HTML property values;
It is easier to absorb / understand a large number in hex than as a long sequence of 1s and 0s;
Easier to write… (if relevant to example)

Other valid examples accepted. Good reason with wrong example not accepted.
R saves space!

1

[7]

Q13.

(a) 0000 0000 0001 1001;

Note: possible use of misprinted scripts: if answers are in the right boxes mark as above. If marks are against the marks allocated, interpret 1st answer as pure binary and 2nd answer as BCD.

1

(b) (i) 53;

1

(ii) 0011 0010; 0011 0101;

2

[4]

Q14.

(a) **Mantissa**

Significant digits/precision/answer by example;

1 mark

Exponent
Power of 2 by which mantissa is to be multiplied to get original value/How many places the point has to move/answer by example;
R decimal point

1 mark

2

(b) (i) Mantissa ↓ Mantissa identified

0110101100 000011

1

(ii) Msb/leftmost bit/starts with determines sign of number;

0 so +ve &/or 1 if –ve.

1

(c) <u>Convert –3 into 2's complement</u>; 0000 0011

Add to 2's complement value of +5; 1111 1101;

If 3-5 calculated correctly give 1 method mark <u>0000 0101</u>+
000 0010;

2

(d) Increased range that can be stored in a given number of bits;

1

[8]

Q15.

(a) 89;

1

(b) Pixel;
Sound;
Instruction/ part of program;
Address / pointer;
Boolean;
1 Unicode character;
2 EBCDIC characters;
Signed integer;
Floating point / real / fixed point / single;
Enumerated type / set;
Status
R flags
R double

Max 3

[4]

Q16.

(i) 10110000

1

(ii) 00110010

1

(iii) 00000001

1

(iv) 10001011

1

[4]

Q1.

(a) Many candidates were able to work out the correct answer to this part showing that they have an understanding of data encoding.

(b) Candidates who answered part (i) generally gave the correct result. ASCII was sometimes given for part (ii) showing that the candidate had not read the question. There continues to be a problem with the spelling of EBCDIC. This is a technical term that should be understood by the candidates and there is no guarantee that misspellings will be given credit in the future.

(c) It was disappointing to see how many candidates were unable to answer this part satisfactorily.

(i) Candidates should be aware that each pixel is stored separately in bit-mapped graphics. Although many candidates stated that the colour would have to be stored, few were able to explain how. A common misconception was that one bit could store a range of numbers.

(ii) There was even less understanding shown of vector graphics. Candidates should appreciate what needs to be stored. Stating that the line would be stored as an equation is insufficient.

Q2.

This question was done very well by a majority of candidates which was very pleasing.

In part (a) a minority of candidates could not convert the binary number 1011 1110 0100 into the hexadecimal equivalent of BE4.

Part (b) of the question asked for the working to be shown, but many candidates did not seem to know what this meant. Putting the place values above the binary pattern would have sufficed. Some candidates did not read the question carefully enough and made the number negative. Some seemed to think wrongly that the place values after the binary point are integer values rather than fractional values.

In part (c) again some candidates did not read the question carefully enough to appreciate that this time the binary pattern represents a negative integer. Some converted the 2’s complement number into its positive equivalent but then forgot to write down the negative sign in front of 1052.

For part (d) again the fact that this was a negative number was forgotten on the way to calculating the answer. Partial marks were awarded where the working was clear and showed that the candidate knew it was a negative number and that the **binary** point would be moved 4 places to the right. Many got confused over what the place values were, in particular, some candidates had difficulty adding the negative whole number part to the fractional part.

It was pleasing to see that so many candidates knew that a normalised negative number starts with 10.

Q3.

(a) Some candidates managed to subtract 18 from 6 rather than 6 from 18, but most managed to use two's complement correctly and so gained at least 3 marks. Some candidates did not take note of the fact that integers were to be stored using 16 bits and so lost the 4th mark.

(b) (i) Most candidates could translate the hexadecimal number A802 into the binary equivalent of 1010100000000010. But hardly any candidates then interpreted this pattern as a floating point number as stated in the question and therefore did not arrive at the correct answer of –2.75. Candidates who did not arrive at the correct answer but showed their working were given credit for correctly identifying the number as being negative and for showing that the binary point moves 2 places to the right.

(ii) Some candidates correctly stated that the reason for storing floating point numbers in normalised form is to maximise precision for a given number of bits. Many candidates however only stated that the reason was to maximise precision, which was not enough to gain credit.

Q4.

It was pleasing to see that most candidates did very well on this question. Part (a) was almost universally correct. Although most candidates gained full marks for parts (b) and (c) there were some candidates who had the answers 256 and 255 reversed.

Q5.

Most candidates correctly stated the hex equivalent as B76. The unsigned fixed point number interpretation was mostly correctly calculated as 183.375, although some candidates managed to write down a negative number instead. Two's complement floating point numbers remain a challenge to a large number of candidates. Partial marks for correct working were awarded. The fact that normalised numbers have a bit pattern with the first two bits different was not widely known. A pleasing number of candidates did know that numbers are stored in normalised form to maximise accuracy and that this also means that a given number can only be expressed in one way with a given number of bits.

Q6.

A significant number of candidates do not know that 1024 bytes make 1 Kilobyte. Even fewer candidates could state correctly that the largest pure binary integer that can be stored in 2 bytes is 65,535 (or 1111 1111 1111 1111 1111). Incorrect responses ranged from as low as 3 to many thousands.

The bit pattern asked for was largely well answered, but candidates should be made aware that leading zeros are required when bit patterns to a specified length are asked for. The whole purpose of binary is that only two states, 0 and 1, can exist.

The concept of parity checking eludes many candidates. Few could explain how a computer system would use a parity bit. The parity bit is set when the character is first generated, by adjusting the parity bit to make the number of 1s even (for even parity). Then the parity is checked at the receiving end and if the parity is now odd an error has occurred. The question clearly stated that in the given computer system the parity bit was the most significant bit of each byte. However, many candidates only looked at the parity across the whole 16 bits. Many candidates were not able to translate the characters 3 and 7 into ASCII codes with the code ranges for digits explicitly provided by the question.

Q7.

Most candidates recognised that the Twos Complement number in part (a) was positive, although the estimate of its size, which depended on their realising that the exponent was negative, was often wildly out. Where candidates failed to convert the second number correctly, credit was given for relevant working. The commonest error was in not appreciating that the binary point originates between the two leftmost digits. A less common, but more dispiriting error, was the use of the exponent as a power of 10.

Processing numbers in fixed point is quicker than in floating point, less processing is required, but ‘calculation is quicker’ was insufficient, and ‘easier to work out’ seemed to show a complete lack of understanding of human computer interaction. A number of candidates said it was easier to understand but this was not an accepted answer. Fixed point representation can give greater accuracy or precision, although many candidates thought otherwise. Thus fixed point representation would be used where maximum precision is required or where the possible range of numbers is small, or of a set format, such as with currency.

Q8.

Although many candidates scored well on this question, others showed a basic lack of understanding of this topic area. For example, part (b) asked ‘With 6 bits of the op code reserved to denote basic machine operations, how many basic machine operations may be coded?’ Incorrect answers included 1, 2, 6, 13, and 63.

Q9.

Nearly all candidates obtained some marks for this question.

(a) Almost all candidates could convert from ‘Pure Binary’.

(b) Nearly all candidates gave the correct bit pattern for 33 but a large number failed to get the parity bit correct (the question setter had ensured that the parity bit had to be set to ‘1’). In part (ii) few candidates realised that the start and stop bits needed to be different and in some cases these were simply left blank. When a binary pattern is asked for, all places must be filled in with either a 0 or a 1. The parity bit must not change; just because start and stop bits are added. These will be stripped off before parity checks are performed at the receiving end.

Q10.

This question was based on section 13.3 of the specification: Data Representation in Computers. The majority of candidates converted from hexadecimal to binary correctly.

Conversion into decimal from two’s complement frequently showed a lack of understanding of that representation. Many candidates did not appear to recognise that the bit pattern represented a negative number, although some stuck a minus sign in front of their answer with no other indication that they had taken it into account. Too many converted the exponent to 3 only to write down 103.

The reasons given for normalising were frequently weak. One good answer was ‘Allows more precise values to be held in the same amount of memory’. Precision/accuracy alone was insufficient. Another reason is that normalising ensures that only one representation of a number is possible. Incorrect answers included that this was the only way to represent negative or decimal numbers.

Q11.

Nearly all candidates scored some marks on this question.

(a) (i) Nearly all gave 54.

(b) (i) The majority correctly identified 4 as the encoded character.

(ii) Depending on the centre, Unicode or EBCDIC were the correct answers given (with some highly original spellings of EBCDIC), while ‘encryption’ and ‘hexadecimal’ were very popular incorrect answers.

(c) (i) Most gained credit with ‘bitmap’ as their answer.

(ii) Nearly all candidates gained at least one mark but many ignored the fact in the question stem that a black-and-white image was to be stored and went into details about storing coloured images. Resolution was also often described which was not asked for here. The description that the image is broken down into pixels, and these are either black or white, that a one could be stored for each white pixel and a nought for each black pixel or vice versa would have gained full marks.

Q12.

This was about number types and number bases. The majority of candidates earned full marks for parts (a) and (b), although a number did not know what an integer was. Most knew the principles of converting from binary to denary and from denary to hex; marks here were lost through silly mistakes. However, the misconceptions as to the role of hexadecimal notation were surprising. The worst, offered by many candidates, was that a large number stored in hexadecimal notation used fewer bits than in binary.

Q13.

(a) The majority of candidates correctly converted 25 to pure binary integer. However, when asked how this would be stored in a 16-bit word, candidates were expected to write down leading zeros to make up 16 bits. To help candidates with this, a box to complete the bit pattern was provided.

(b) (i) Even with the clue in the question a large number of the candidature did not give 53 as the ASCII code for 5. A very common wrong answer was 85.

(ii) Few included the pair of 11s in the left hand nibble of each number character. The right hand side of each pattern was correct but obviously incomplete, i.e. 0000 0010 0000 0101. Again, a similar question had been set in previous papers.

Q14.

(a) Candidates found it very difficult to explain the terms mantissa and exponent in this part. Many answers were weakened by reference to a decimal point.

(c) Here, it was disturbing that some candidates interpreted ‘subtract 3 from 5’ as 3-5. Candidates who showed their calculation in 3 bit binary were apparently showing the addition of two negative numbers, which was not correct.

(d) In this part the advantage of floating point representation over fixed point representation is the increased range of numbers that can be represented in a given number of bits, not the accuracy of the number.

Q15.

Some centres had not covered this part of the syllabus and the question was either well answered by the majority of the candidates in the centre or answered with wild guesses. In part (b) candidates often gave very vague answers, such as EBCDIC, rather than the more precise ‘2EBCDIC characters’, as was hinted by the information given in the question. Although “bit map”, “image” and similar responses were accepted this time, in future candidates will be expected to give more precise information such as “one pixel” could be stored in a 16 bit word.

Q16.

Those candidates that were prepared for this question usually got full marks. The XOR operation proved the most difficult.