5.2 Number bases Mark Scheme

Q1.

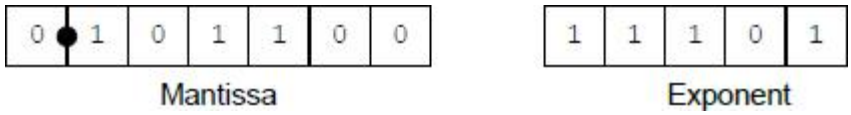
(a) **Mark is AO2 (apply);**

30A;

R. More than one lozenge shaded

1

(b) **All marks AO2 (apply)**



1 method mark for either:

- showing correct value of both mantissa and exponent in decimal (mantissa = 0.6875 // 11/16, Exponent = −3)
- showing binary point shifted 3 places to left in binary number
- indicating that final answer calculated using answer = mantissa × $2^{exponent}$

1 mark for correct answer

Answer = 0.0859375 // 11/128

If answer is correct and some working has been shown, award two marks, even if working would not have gained credit on its own.

2

[3]

Q2.

0111 1011;

[1]

Q3.

256 // 2^8;

[1]

Q4.

7;B;

[2]

Q5.

Easier to read/understand; **R**. easier for computers to read/understand
(Can be displayed using) fewer digits;
More compact when printed/displayed;
NE. Takes up less space

NE. More compact
MAX 1

[1]

Q6.

(a) **All marks AO2 (apply)**

1 mark for working: conversion of D to 13 or multiplication of a number (even if not 13) by 16 and adding 6 to the result;
1 mark for answer: 214;

2

(b) **All marks AO2 (apply)**

1001; 0110;
1 mark: correct first four bits
1 mark: correct bits in position 5 – 8

2

(c) **All marks AO2 (apply)**

1;0111101;
2 marks: Correct answer only

2

(d) **Mark is for AO2 (apply)**

10101011;

1

(e) **Mark is for AO1 (understanding)**

The result is too large to be represented;
(it causes) overflow;
The result represents a negative value;
Max 1 mark

1

[8]

Q7.

(a) 167;;

If final answer is incorrect **Max 1** can be awarded for some correct working out being shown by the candidate:

1010 0111;
10 * 16 // 160 // A * 16;
A = 10;
Multiplying a value by 16 and adding on 7;

2

(b) 0111.1010 // 01111010

Mark as follows:
4 bits before binary point are 0111;
4 bits after binary point are 1010;

2

(c) 1;110 1110;
R if not 8 bits

2

(d) 127;

1

(e) The number to subtract is converted into a negative number;
NE Convert into two□s complement
This is then added to the first number;

Two marks for example:

23 = 00010111
-48 = 11010000;

11100111; (= -25)

A if not used 8 bits in examples
A 23 + - 48 is worth 1 mark only (if there is no description)

Note: for the first mark in the example to be awarded the two bit patterns must be correct. For the second mark in the example accept an incorrect answer as long as it is a correct addition using one of the two correct bit patterns.

4

(f) 11101110;
R 01110111

1

(g) 11101011;
DPT A. 11010111

1

(h) Get the two's complement (of a positive binary value) //
Converts a positive binary value into its negative equivalent;
A It inverts all bits after the first 1 is received;

1

(i)

Input	Original State	Output	New State
0	S0	0	S0
1	S0	1	S1
0	S1	1	S1
1	S1	0	S1

Mark as follows:
S0 as original state for 2nd row;
1 as output for 3rd row;
Final row correct;

3

[17]

Q8.

(a) 0111 1011;

1

(b) 256 // 2^8;

1

(c) 7;B;

2

(d) Easier for people to read / understand;
(Can be displayed using) fewer digits;
More compact when printed/displayed;
NE Takes up less space
NE More compact

Max 1

[5]

Q9.

(a) 167;

1

(b) $10.4375 // 10\frac{7}{16}$;;

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

2

(c) -;89;

1 mark for correct sign
1 mark for correct integer value

2

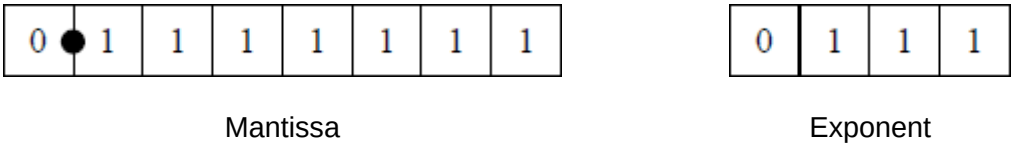
(d) A7;

1

[6]

Q10.

(a)



1 mark for correct bit pattern in both mantissa and exponent.

1

(b) Mantissa = –0.6875 // –11/16
Exponent = 3
Answer = –5.5 // –5½

1 method mark for either:

- showing correct value of both mantissa and exponent in denary
- showing binary point shifted 3 places to right within a correct binary pattern*
- indicating that final answer calculated using answer = mantissa x $2^{exponent}$ (**A** mantissa in denary or binary but exponent must be in denary)

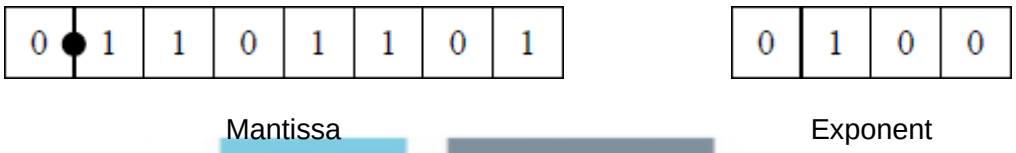
1 mark for correct answer

* Correct binary patterns with the binary point shifted 3 places are:

1010.1000	0101.1000
1010.1	101.1000
	101.1

2

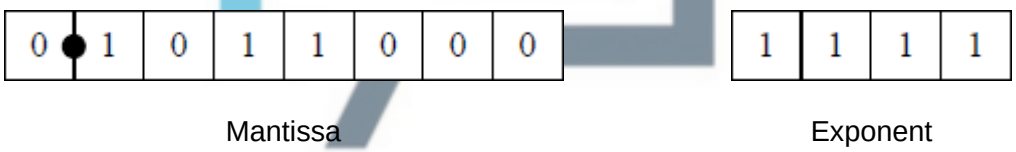
(c)



1 mark for correct mantissa
1 mark for correct exponent

2

(d)



1 mark for correct mantissa
1 mark for correct exponent

2

(e) **Definition** *(2 marks):*
The result of a calculation is too large to store/represent // a number is too large to store/represent;
In the available number of bits / storage space (allow example e.g. data type, byte, word, example of a data type);
R space **NE**

Example *(1 mark):*
Multiplying two numbers together;
Dividing a number by a number less than one / small number;
R zero
A Adding two numbers (of same sign)
A When number converted from one type to another that does not have suitable range / enough bits / enough storage space to represent it
A Answers by example

Max 1

3

[10]

Q11.

(a) 984;

1

(b) 984;

1

(c) (i) – 13.0;;;
Allow method marks
1 mark for correctly identifying negative number
1 mark for integer value correct
1 mark for fraction (dependant on correct integer value)// 01101.000

3

(ii) To Maximise precision in a given number of bits //
To minimise rounding errors;
A to Maximise accuracy in a given number of bits

1

(iii) leftmost 2 digits/bits are different//
a significant bit is stored after the binary point//
bit after point different from bit before point;
A the first bit after the sign bit is a '0';
A The second bit is a '0';
A *an answer that clearly implies a '0' follows the '1'*

1

(iv) 127// $2^7 - 1$;;
Max 1 for correct mantissa (01111111) or exponent (0111/7)

2

[9]

Q12.

(a) 255 / 11111111 / FF / 2^8 – 1 ;

1

(b) (i) 81 ;

1

(ii) 1000 1010 ;

1

(c) 522

1

(d) (i) 100 0010 ;
R leading 0 as 8th bit

1

(ii) The total is calculated before transmission ;
0 or 1 bit is added (to the 7-bit code) ;
So that, the total number of 1 bits must compute to an even number ;
The number of one bits is re-calculated after received ;
An odd number of 1 bits indicates an error ;

Max 2

[7]

Q13.

Address 56 - 55 ;

Address 57 – 137 ;

[2]

Q14.

(a) E X A M ;

Mark as follows:
1 or 2 correct (1) ;
3 correct (2) ;
4 correct (3) ;
R lower case

3

(b) (i) Universal Serial Bus;

1

(ii) Parallel ;

1

(iii) Set of rules ;
Sending signals between devices;
(Computer) asks are you ready? ;
(Printer) acknowledges yes I am ;
(Computer) responds here comes the data ;
(Printer) 'thank you received' ;

Max 2

(iv) Acknowledge data received by the printer ;
Error ;
Line is busy / free / ready /'status' / ACK Request ;
Timing / strobe;
Interrupt;
R Ground

Max 1

(v) Operating system ;
Word processing software / text editing software / any sensible Application ;
Print spooler ;
Printer driver ;
R 'printing software'

Max 2

[10]

Q15.

(a) 00011011;
1;1101101;

3

(b) 10000;1000; Allow FT from (a)

Stop marking when an error occurs

2

(c) (i) Carry Bit;
R Overflow bit

1

(ii) For multi-word/byte operations//<u>test</u> for overflow

1

(d) 1B;
ED;
Allow FT

2

[9]

Q16.

(a) (i) **Do not mark this part**

(ii)

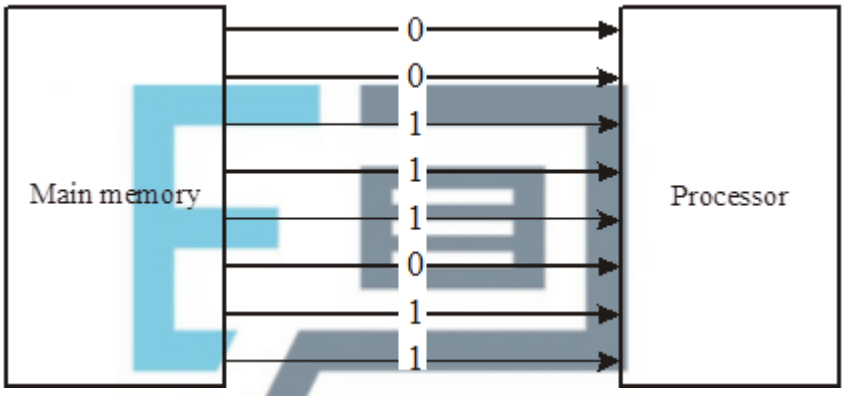


Diagram has 8 lines;
Diagram has the correct 1 bit per line; A**.** pulses to indicate 0/1's

2

A either 'top to bottom' or 'bottom to top' labelling of the bits
I. the direction of any arrows

(b) Interpretation:

- program instruction/command;
- character / ASCII code / 7 data bits + parity bit;
- integer / 59 / number;
- real / number;
- byte/pixel from a graphics file; R. 'part of
- byte/sample from a sound file; R. 'part of
- an address;

R BCD digits

Max 3

[5]

Q17.

(a) 974;

1

(b) 151.25;;

1 mark for integer part, 1 mark for fractional part

2

(c) –104.75;;

If answer not correct award 1 mark for attempt at complementing the binary pattern

2

(d) (i) –13.125;

Allow method marks
1 mark for 2 4 seen or correct 4 bit shift
1 mark for integer value correct including sign
1 mark for fractional part

3

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

1

[9]

Q18.

(a) 140 ¼ ;;

1 mark for correct integer part,

140.25;;

1 mark for correct fractional part

2

(b) (i) -14.5;;;

Give 2 marks for 14.5
Partial marks for workings if result incorrect:
1 mark for negative number;
1 mark for $x2^4$ (accept 16 instead of 2^4);
A showing that binary point moves 4 places right;

3

(ii) Leftmost 2 digits/bits are different;
A significant bit is stored after the binary point;
Bit after point different from bit before point;
(negative number) starts with 10…
(positive number starts with 01)….;
A the first but after the sign bit is a '0';
A The second bit is a '0';
A an answer that <u>clearly</u> implies a '0' follows the '1'

Max 1

(iii) To maximise accuracy/precision <u>for a given number of bits</u>
// to minimise rounding errors;
A more accurate/precise for <u>a given number of bits</u>;
A given number can only be expressed in one way <u>in a given number of</u>

bits
// a given number can only be expressed in one way in a given format;
To simplify arithmetic/logical operations;
I range

1

[7]

Q19.

(a) 1101 0001 0101 1010;;

1 mark for each correct byte
I leading bits

2

(b) 16;

1

[3]

Q20.

(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]

Q21.

(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 <u>in a given number of bits</u> // to minimise rounding errors // to have just one representation of the decimal number // to simplify arithmetic operations;
A to maximise accuracy <u>in a given number of bits</u>;

1

[9]

Q22.

(i) 56,576 / $13 \times 16^3 + 13 \times 16^2$ (*as final answer*);

1

(ii) 16;

1

(iii) 16;

1

(iv) $512_{(10)}$; 2^9; **A** 200_{16}; **A** &200;

1

[4]

Q23.

(a) It calls itself / is defined in terms of itself / is re-entrant / contains within its body a reference to itself;

Ensure 'it' refers to procedure, if meaning program or object no mark

1

(b) The current state of the machine must be saved/preserved so can return correctly to previous invocation of B;

OR
Return address / procedure parameter / status register / other register values / local variables must be saved/preserved so can return correctly to previous invocation of B**)**;

1

(c)

Call Number	Parameter
1	53
2	26
3	13
4	6 ;
5	3 ;
6	1 ;

Printed Output: 1 1 0 1 0 1;;;

1 mark for each correct pair of bits
Mark from left and stop marking when error encountered ignore punctuation. if more than 6 bits give a max of 2 marks

6

(d) Conversion (of a denary number) into binary;

1

[9]

Q24.

(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]

Q25.

(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]

Q26.

(a)

y	x	Index	Result		
			[3]	[2]	[1]
–	5	0	–	–	–
1	2	1	–	–	1
0	1;	2;		0;	
1	0;	3;	1;		

Mark for x=1
Mark for index=2
Mark for y=0 AND Result[2]=0
Mark for x=0

Mark for index=3
Mark for y=1 AND Result[3]=1
Ignore other cells

6

(b) Convert an integer into its binary equivalent;

1

[7]

Q27.

(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]

Q28.

(a) 4E

1

(b) 78

1

[3]

Q29.

(a) 111101

(b) 3D

[2]

Examiner reports

Q1.

(a) Program that calculated the score for a word – if done correctly full marks could be obtained by doing this but a number of students made mistakes when writing their own score-calculating code.

(b) This question part was satisfactorily tackled but only approximately half of students got both marks. Common mistakes were to shift the binary point in the wrong direction or to treat the exponent as an unsigned integer instead of as a two's complement value.

Q7.

The topics covered by this question were generally well-understood. Most students were able to answer parts (a)-(c) well, with the most common error being not reading the question carefully and using a different number of bits from that specified. For part (d), a significant number of students seemed to have memorised that 255 was the largest positive number that can be represented using 8-bit unsigned binary integers and gave this as their answer, even though the question was about two's complement binary.

Answers to part (e) showed that a number of students were not familiar with binary subtraction. However, students who knew how two's complement can be used to subtract one number from another were normally able to describe the process clearly and get good marks. Some answers seen made it clear that students thought that a two's complement binary number means a negative number, not understanding that the two's complement system can be used to represent both negative and positive numbers.

Parts (f)-(i) about finite state machines with output were not answered well with a number of students unable to work out the correct output strings even though there were two completed examples given in the question. Most students were able to get some marks for completing the state transition table in part (i) but few obtained full marks.

Q8.

Most candidates obtained the mark for part (a). Those who did not get the mark often used 7 bits instead of 8 – 1111011 – (missing out the 0 on the LHS) or miscounted the number of 1s to use – 0111011. Part (b) was generally well-answered. The most common incorrect answers were 255 (the highest decimal value that can be represented using 8 bits) and 128 (the number of different values that can be represented using 7-bit binary). This suggests that candidates are recognising that they need to do 2^8, but often then get confused about when they need to take a 1 away – getting muddled finding the highest value with finding out the number of values that can be represented. Some candidates are taking 1 away from 2^8 and some are taking 1 away from 8 and giving an answer of 2^7. A larger number of candidates got the correct answer to the hexadecimal question this year, but a significant proportion did not understand what hexadecimal is and made either no attempt or gave an answer that was not in hexadecimal. Answers for part (d) often lacked precision and did not convey the idea that it is easier for a person to read the hexadecimal equivalent of a binary bit pattern. A common misconception is that hexadecimal values will use less storage space.

Q9.

Most candidates were able to answer this question. If mistakes were made they tended to

be on the two's complement and hexadecimal questions. Some candidates wrote "10 7" as an answer for part (d) – the bits had been split into blocks of 4 but then converted into decimal values.

Candidates should be encouraged to check their answers carefully as marks were sometimes dropped due to arithmetic errors.

Q10.

Part (a): Most candidates were able to write correctly the largest positive number that could be represented in the given normalised floating point system.

Part (b): This question part was well answered, with many candidates getting both marks. The most common error was to forget that the most significant bit of the mantissa had a negative value, resulting in an answer of 10.5 rather than -5.5.

Part (c): The majority of candidates got full marks for this question part. Many of those who failed to do this clearly knew the method to use, but made arithmetic errors.

Part (d): This question part was well answered, with many candidates correctly calculating both the mantissa and exponent. A small number calculated the mantissa correctly, but failed to normalise this properly, resulting in an incorrect value for the exponent.

Part (e): This question part was very well answered. Most candidates correctly explained that overflow occurs when the result of a calculation is too large to store in the available number of bits. Many were also able to give an appropriate example of how this might occur.

Q11.

A large number of candidates were clearly not well-prepared for what is intended to be a straight forward starter question. In part (c) many candidates were unable to complement the negative mantissa. The role of the exponent was generally understood but converting the resulting bit pattern into a decimal value was not. Parts (c)(ii) and (c)(iii) should have been standard bookwork, but they presented a real challenge with few candidates obtaining full marks. Part (c)(ii) presented a particular challenge. In part (c)(iv) there were far too many answers of 255 or 256 showing a lack of understanding.

Q12.

Parts (a) (b) and (c) were often poorly answered which is at odds with the general level of answers seen on previous papers. Was it because the computations were put into a context that students were unable to identify with?

(d) (i) Some candidates did not read the rubric and wrongly gave an 8-bit code.

(ii) Again the candidates' communication skills often let them down. Their answer was often sufficient to suggest to the examiner that the candidate probably did know how parity worked but they were unable to express the key parts of the process – there were many points the candidate could have described to secure the two available marks.

Q13.

By and large was well answered with most candidates scoring the full 2 marks.

Q14.

(a) The majority of candidates scored the full 3 marks.

(b) (i) A surprising number of candidates did not score marks on this question. There were many different wrong answers including, for example, “Ultra Slim Build” and “Uniform Byte Synchroniser”.

(ii) Most correctly stated parallel.

(iii) There were a variety of ways the candidate could score the 2 marks. For example, by focussing on the word, protocol and describing this as a set of rules for communication. The most common answers gave particular signals which are exchanged between the two devices.

(iv) There were few correct answers seen here despite an exhaustive list of possibilities on the mark scheme. Many candidates confused this scenario with the use of signals on the control bus of the motherboard.

(v) This was poorly answered. The vague term ‘printer software’ was not considered acceptable. Printer driver was common from the stronger candidates together with “word processing software” or even “the applications program from which the document is being printed”. Few mentioned the operating system. Other common wrong answers were the suggestion that the data file to be printed was software, or describing the ASCII code table (referring back to part (a) of the question) as software.

Q15.

Most candidates were able to convert 27 into binary. Fewer candidates were able to deal with a negative number satisfactorily. Many candidates who failed to obtain the correct value for –19 did not add together their binary values but simply converted 8 into binary for their answer to part (b). Most of the candidates who managed to obtain the correct answers for part (a) also obtained the correct answer for part (d). From time to time in part (d) the correct answers for 27 and -19 were given in hexadecimal although incorrect binary patterns were provided in part (a). This suggested, as did a number of the other incorrect responses that were seen, that some candidates attempted to use electronic calculators to find the answers to question one without really understanding what they were being asked to do.

It was very disappointing to see how few candidates had any clear idea of the name or the use of the additional bit.

Q16.

(a) Due to an error in the question paper for part (i), examiners were instructed not to mark this part question. The total mark for the paper was therefore reduced to 64. Despite a different style of question for part (ii), the vast majority of candidates were clear as to what was required. Common errors were a diagram which showed 9 lines instead of 8 or the lines drawn as ‘tubes’. Some candidates drew pulses on the lines instead of labelling each one with a 1 or 0 and still scored both marks.

(b) This was surprisingly badly answered with candidates either not clear as to what was being asked for, or not understanding the phrase “data representation”. From past examination paper responses this appears to have been well understood previously. If the candidate has covered the specification content fully then it is hoped that students would be fascinated to learn how the same binary number can be interpreted in so many different ways: a basic machine code instruction to add

two numbers together, a musical note, the colour of a pixel, one of the characters in a text string, one of several different number types etc.

Q17.

This question gave most candidates a good, often very good, start to the paper. Surprisingly, the floating-point question (part (d)(i)) was answered correctly more often than the fixed point one (part (b)). Some candidates lost marks for partially correct answers through not showing all of their working.

Part (b)(ii) was not well answered. Many candidates mentioned improved precision or accuracy but failed to state that this also depends on the number of bits used for the representation.

Q18.

(a) It is very worrying that many candidates gave a negative answer when the question clearly stated that the bit pattern represented an unsigned number.

(b) (i) Two's complement is largely understood but some candidates missed out on marks by not showing working and getting their final answer incorrect, so gaining no marks. Part marks were awarded for showing a correct method of conversion.

(ii) Many candidates correctly stated that the bit pattern starting with 10 showed that the number was normalised, but some candidates, incorrectly, stated that it was because there were no leading zeros, clearly not appreciating that a negative number will always start with a 1 but may not be normalised.

(iii) Many candidates, correctly, stated that normalised form would maximise precision with a given number of bits.

Q19.

(a) Most candidates gained full marks for converting the hexadecimal digits to binary.

(b) However, many candidates could not see that the address bus would require 16 lines to convey the resulting 16-bit number.

Q20.

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.

Q21.

(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.

Q22.

This question was not very well answered by the vast majority of candidates. Most had great difficulty already translating &DD00 into denary. The answer 56576 gained the mark as did $13 \times 16^3 + 13 \times 16^2$ as a final answer. Few candidates could see that for a 4-digit hex number 16 bits are required to represent this number in binary. Even fewer could make the connection that 16 address lines were therefore required. Many candidates seem to be under the illusion that one line can carry 8 bits. Calculating the number of data cells from addresses &DD04 to &DF03 also caused much difficulty. Those candidates that got anywhere near the correct answer, subtracted one from the other, but forgot to add on one to get to the correct answer of 512 or 200_{16}.

Q23.

Those candidates who clearly understood recursion scored high marks in this question, but a worrying number of candidates could not explain that a procedure is recursively defined when it is defined in terms of itself. A stack is needed so that register values such as return address and parameter values can be saved and can be returned to in the correct order. Most candidates managed to complete the trace table correctly but many did not give the correct printed output. Many candidates did not provide the correct number of digits, or in reverse order. A large majority wrongly thought that procedure B described a binary search. Interestingly, some of the candidates who got the printed output wrong still stated the correct purpose of the procedure: converting the denary number provided as parameter into binary.

Q24.

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.

Q25.

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.

Q26.

Candidates find following pseudo-code very difficult and very few were able to complete the table correctly, though many got some parts correct – usually the next values of x and Index. Of those who did complete the table correctly only a few spotted that the initial value of x (i.e. 5) had been converted into binary.

Q27.

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.

Q28.

If candidates were prepared for this question they generally produced a completely correct solution. Some candidates were unprepared and failed to attempt the question at all.

Q29.

Most candidates were able to convert the value to binary but far fewer were able to successfully convert the value to hexadecimal.