Past-paper walk-through

1.1.1 ▪ 1.1.1

eXercise

Questions in this set

- 9618/11 N25 Q1 ▪ 7 marks
- 9618/13 N25 Q2 ▪ 6 marks
- 9618/12 J25 Q2 ▪ 7 marks
- 9618/13 N24 Q8 ▪ 6 marks
- 9618/11 J24 Q7 ▪ 3 marks
- 9618/12 J24 Q7 ▪ 6 marks

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 1

9618/11 N25 ▪ Q1 ▪ 7 marks

(a)(i) Convert the binary number into hexadecimal.

101100111010 **[1]**

(a)(ii) Convert the denary number into Binary Coded Decimal (BCD).

108 **[1]**

(a)(iii) Convert the 12-bit two's complement binary integer into denary.
Show your working.

111110111100 **[2]**

(b)(i) Complete the calculation using binary addition.

$$\begin{array}{r} 10110011 \\ +01111000 \\ \hline \end{array}$$

Question 1

[1]

(b)(ii) Name and describe the error that can occur when binary addition is performed. **[2]**

Solution 1

(a)(i) Worked steps

Hexadecimal is base 16 and binary is base 2, so **four** binary digits make exactly one hex digit. Group from the right.

- `1011 0011 1010`
- $1011 = 11 =$ B, $0011 = 3$, $1010 = 10 =$ A
- Answer: `B3A`

Solution 1

Grouping from the LEFT is the classic error: pad the front with zeros instead, so the last group is always complete.

Mark scheme

- B3A

Solution 1

(a)(ii) Worked steps

BCD stores each **denary digit** separately in four bits — it is not the binary value of the whole number.

- 108 is the digits 1, 0, 8.
- $1 \rightarrow 0001$, $0 \rightarrow 0000$, $8 \rightarrow 1000$
- Answer: `0001 0000 1000`

Solution 1

Compare with plain binary, where $108 = 1101100$ — quite different, and that contrast is what the question tests.

Mark scheme

- 0001 0000 1000

Solution 1

(a)(iii) Worked steps

The leading bit is 1, so the number is negative. Flip and add one to read its magnitude.

- 1111 1011 1100 → flip → 0000 0100 0011 → add 1 → 0000 0100 0100 = 68
- So the value is -68.

Or add the place values directly with the top bit negative:
$-2048 + 1024 + 512 + 256 + 128 + 32 + 16 + 8 + 4 = -68$. Either method scores; show whichever you use, because a mark is for the working. **Mark scheme**

- 1 mark for the working
- 1 mark for the correct denary value

Solution 1

- Working:
- Method 1:
- 1111 1011 1100
- Flip the bits 0000 0100 0011
- Add 1
- 0000 0000 0001 +
- 0000 0100 0100
- Method 2:
- -2048 + 1024 + 512 + 256 + 128 + 32 + 16 + 8 + 4
- // -128 + 32 + 16 + 8 + 4

Solution 1

- Denary value:
- -68

Solution 1

(b)(i)

Worked steps

- Add the columns from the right, carrying as usual: $10110011 + 01111000 = 1\,00101011$.
- Only eight bits are stored, so the answer kept is `00101011`.

Mark scheme

- (1) 00101011

Solution 1

(b)(ii)

Worked steps

- The error is **overflow**.
- The true answer needs nine bits but the register has only eight, so the carry out of the most-significant bit is lost and the stored result is wrong.

Name it **and** describe it: the two marks are for the name and for what goes wrong.

Mark scheme

- 1 mark per bullet point, max 2 marks
- ▪ An overflow error occurs
- ▪ The answer cannot be represented in the number of bits available // The answer is larger than the maximum positive number that can be stored in the register // The answer is smaller than the most negative number that can be stored in the register

Question 2

9618/13 N25 ▪ Q2 ▪ 6 marks

Data in a computer system is represented in binary.

(a) Put one tick (✓) in each row to identify the minimum number of bits used to store each example of data.

Example of data	**Number of bits**						
	4	**8**	**16**	**24**	**32**	**64**	**128**
the hexadecimal value F139							
16 000 000 unique amplitude values							
an IPv4 address							
256 unique colours							
an IPv6 address							
the denary value 65 000							

Question 2

[3]

Question 2

Example of data	Number of bits						
	4	8	16	24	32	64	128
the hexadecimal value F139							
16 000 000 unique amplitude values							
an IPv4 address							
256 unique colours							
an IPv6 address							
the denary value 65 000							

Question 2

9618/13 N25 ▪ Q2 ▪ 6 marks

Data in a computer system is represented in binary.

(b) Convert the denary number -108 into a 12-bit two's complement binary number. **[1]**

Question 2

9618/13 N25 ▪ Q2 ▪ 6 marks

Data in a computer system is represented in binary.

(c) A three-place arithmetic shift to the right is performed on the following two's complement negative integer.

Show the result of this arithmetic shift.

10010011 **[1]**

Question 2

9618/13 N25 ▪ Q2 ▪ 6 marks

Data in a computer system is represented in binary.

(d) Convert the following positive binary integer into hexadecimal.

1110001100111011 **[1]**

Solution 2

(a) Mark scheme

- 3 marks for 6 correct ticks
- 2 mark for 4 or 5 correct ticks
- 1 mark for 2 or 3 correct ticks
- Example of data
- Number of bits
- 4
- 8
- 16

Solution 2

- 24
- 32
- 64
- 128 the hexadecimal value F139
- ✓
- 16 000 000 unique amplitude values
- ✓ an IPv4 address
- ✓
- 256 unique colours
- ✓ an IPv6 address

Solution 2

- ✓ the denary value
- 65 000
- ✓

Solution 2

(b) Worked steps

Two's complement in three steps, always the same three.

1. Write the **magnitude** in the required width: $108 = 64 + 32 + 8 + 4$, so in 12 bits `0000 0110 1100`.
2. **Invert every bit**: `1111 1001 0011`.
3. **Add one**: `1111 1001 0100`.

Solution 2

Check it in one glance: the leading bit is 1, which is what a negative number must look like. A quick verification is to add the answer to the positive form — the 12 bits should come out all zero, with the carry falling off the end.
The width is part of the question. Twelve bits means twelve digits, so keep the leading zeros in step 1 or the inversion is wrong.

Mark scheme

- 1 mark
- 1111 1001 0100

Solution 2

(c) Worked steps

An **arithmetic** shift preserves the sign, which is the whole difference from a logical one: the bits vacated on the left are filled with copies of the **sign bit**, not with zeros. Starting from 10010011, whose sign bit is 1, shift right three places filling with 1s:

- one place: 11001001
- two places: 11100100
- three places: 11110010

Solution 2

So the answer is `1111 0010`.
Each right shift halves the value, and the number stays negative — filling with zeros would turn a negative number positive, which is exactly the mistake the word *arithmetic* is there to prevent.

Mark scheme

- 1 mark
- 1111 0010

Solution 2

(d) Worked steps

Hexadecimal is base 16 and binary is base 2, and $16 = 2^4$ — so each hex digit is exactly **four** binary digits. Group from the RIGHT and convert each group on its own:
`1110 0011 0011 1011`

- $1110 = 14 =$ **E**
- $0011 = 3$
- $0011 = 3$
- $1011 = 11 =$ **B**

Solution 2

The answer is `E33B`.
Sixteen bits give exactly four groups here, so nothing is left over. When the bit count is not a multiple of four, pad on the LEFT with zeros — padding on the right multiplies the number.

Mark scheme

- 1 mark
- E33B
- 1
- October/November
- 2025

Question 2

9618/12 J25 ▪ Q2 ▪ 7 marks

(a) Convert the denary integer 558 into 12-bit binary and hexadecimal. **[2]**

(b)(i) Convert the two's complement binary integer 11100010 into denary. **[1]**

(b)(ii) Write the smallest and the largest two's complement binary integers that can be represented in 8 bits. **[2]**

(c) Give **one** application where Binary Coded Decimal (BCD) is used and justify its use. **[2]**

Solution 2

(a) Worked steps

Do the binary first, then read the hexadecimal straight off it — converting twice from denary is twice the work and twice the risk.
Subtract the largest power of two that fits, repeatedly:

- $558 - 512 = 46$ (bit for 2^9)
- $46 - 32 = 14$ (bit for 2^5)
- $14 - 8 = 6$ (bit for 2^3)
- $6 - 4 = 2$ (bit for 2^2)
- $2 - 2 = 0$ (bit for 2^1)

Solution 2

That gives 10 0010 1110, padded to twelve bits as 0010 0010 1110.
Now group those same twelve bits in fours: 0010 = 2, 0010 = 2, 1110 = E, so the hexadecimal is 22E.
Check by adding the values back: $512 + 32 + 8 + 4 + 2 = 558$.

Mark scheme

- 1 mark for:
- 0010 0010 1110
- 1 mark for:
- 22E

Solution 2

(b)(i) Worked steps

The leading bit is 1, so 11100010 is negative and the magnitude has to be recovered — you cannot simply add up the place values.
Reverse the two's complement procedure:

1. **Invert every bit**: 00011101.
2. **Add one**: 00011110.
3. Read that as an ordinary binary number: $16 + 8 + 4 + 2 = 30$.
4. Put the sign back: the answer is -30.

Solution 2

The same answer comes from treating the top bit as -128: $-128+64+32+2=-30$. Either method earns the mark; the second is quicker once you trust it.

Mark scheme

- 1 mark for:
- -30

Solution 2

(b)(ii)

Mark scheme

- 1 mark for smallest, 1 mark for largest
- Smallest: 1000 0000
- Largest: 0111 1111

Solution 2

(c) Mark scheme

- 1 mark for application and 1 mark for corresponding justification e.g.
- ▪ an application that performs financial / banking calculations
- ▪ because financial transactions use only two decimal places and must be accurate, no accumulating / rounding errors and it is difficult to represent decimal values exactly in normal binary.
- Or
- ▪ electronic displays
- ▪ because visual displays only need to show individual digits

Solution 2

- ▪ because conversion between denary and BCD is straightforward
- Or
- ▪ The storage of the date and time in the BIOS of a PC
- ▪ because conversion between denary and BCD is more straightforward
- Or
- ▪ Barcode systems
- ▪ because conversion between denary and BCD can be accurately completed

Question 8

9618/13 N24 ▪ Q8 ▪ 6 marks

8 **(a)** Convert the hexadecimal number 1FAB into denary.

Question 8

(b) Explain how to convert the two's complement integer 10011111 into denary. Give the denary value after conversion.

Question 8

[3]

(c) Describe the difference between a right logical binary shift and a right arithmetic binary shift.

Solution 8

(a) Worked steps

Expand in base 16. Write the place values above the digits first — they are 16^3 down to 16^0, that is 4096, 256, 16 and 1.

- `1FAB`: $1 \times 4096 + 15 \times 256 + 10 \times 16 + 11 \times 1$
- $= 4096 + 3840 + 160 + 11$
- $= \mathbf{8107}$

The letters are where this goes wrong: `F` $= 15$, `A` $= 10$, `B` $= 11$. The digits run A $= 10$ through F $= 15$, so a quick way to check one is to count up from A.

Solution 8

Show the working. One mark is available and the question asks for it, so the four products should appear on the page even though only the total is being marked.

Mark scheme

- 1 mark for:
- Denary value: 8107
- 1
- 9618/13
- October/November
- 2024

Solution 8

(b) Mark scheme

- 1 mark for each bullet point for the method (max 2)
- e.g.
- ▪
- Flip each bit then add 1 ...
- ▪
- ... method of converting the new binary number into denary
- ▪
- Most significant 1 bit is treated as the corresponding negative

Solution 8

- denary value ...
- ▪
- ... add the other positive corresponding denary values
- 1 mark for correct conversion
- Denary value: -97
- 3

Solution 8

(c) Mark scheme

- 1 mark each:
- ▪
- A logical shift moves all bits to the right and inserts zeros in the
- appropriate leftmost bits
- ▪
- An arithmetic shift moves all bits to the right but copies the sign bit into
- the Most Significant Bit (MSB)
- 2

Question 7

9618/11 J24 ▪ Q7 ▪ 3 marks

7 Complete the binary addition. Show your working.
1 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1

- 0 0 0 1 1 0 0 1

[3] **[3]**

Solution 7

Worked steps

Add the columns from the right, carrying into the next column exactly as in denary — but carrying at 2 rather than at 10.

```
      1 0 0 1 1 1 1 0
      0 1 1 0 0 0 0 1
  +   0 0 0 1 1 0 0 1
  ------------------
 (1) 0 0 0 1 1 0 0 0
   1 1 1 1 1 1 1        (carries)
```

Solution 7

Three marks, and each is for a different thing:

1. **The working** — the carried values written in, clearly, under or above the columns.
2. **The answer**, `0001 1000`.
3. **The overflow**, clearly labelled as an overflow.

That third mark is the point of the question. The true total needs nine bits, and only eight are available, so the leading 1 has nowhere to go: the register holds `0001 1000` and a flag records that the result is wrong. Writing the ninth bit into the answer, or quietly dropping it, both lose the mark — one gives a nine-bit answer, the other hides the error.

Solution 7

Check the arithmetic in denary: $158 + 97 + 25 = 280$, and $280 - 256 = 24$, which is 0001 1000. The 256 that was subtracted is precisely the overflow. **Mark scheme**

- 1 mark each:
- ▪
- Working – carried values clearly indicted
- ▪
- Correct answer 0001 1000
- ▪
- Overflow clearly indicated as overflow
- Example:
- 1 0 0 1 1 1 1 0
- 0 1 1 0 0 0 0 1
- + 0 0 0 1 1 0 0 1
- (1) 0 0 0 1 1 0 0 0
- 1 1 1 1 1 1 1.....(carries)

Question 7

9618/12 J24 ▪ Q7 ▪ 6 marks

7 A computer stores binary data.

(a) Tick (✓) **one** box only to identify the **largest** file size.

- ☐ 3300 kibibytes
- ☐ 0.3 megabytes
- ☐ 3 mebibytes
- ☐ 3300 kilobytes

[1]

(b) Subtract the denary number 10 from the denary number 100 using binary subtraction.

Question 7

Show your working.

Question 7

Answer .....

[3]

(c) Convert the hexadecimal number C0F into denary.

Show your working.

Question 7

Answer .....

[2]

Solution 7

(a)

Mark scheme

- 1 mark for:
- 3300 kibibytes
- 1

Solution 7

(b) Worked steps

Computers do not subtract; they **add the negative**. So convert both numbers, negate the one being subtracted, and add.

- $100 = 64 + 32 + 4$, so 0110 0100.
- $10 = 8 + 2$, so 0000 1010.
- Negate 10 in two's complement: invert to 1111 0101, add one $\rightarrow$ 1111 0110.
- Add:

```
   0 1 1 0 0 1 0 0        (100)
+  1 1 1 1 0 1 1 0        (-10)
 -----------------
(1) 0 1 0 1 1 0 1 0
```

Solution 7

- The carry out of the top bit is **discarded**, leaving `0101 1010`.
- Check: $64 + 16 + 8 + 2 = 90$, and $100 - 10 = 90$. ✓

Three marks: both conversions to binary, the method (negate and add, or a correct direct subtraction), and the right answer.

Solution 7

The carry out is not an overflow here. In two's complement addition of numbers with opposite signs a carry out of the most significant bit is normal and is thrown away — overflow only occurs when two numbers of the *same* sign produce a result of the other sign. **Mark scheme**

- 1 mark each:
- ▪
- Converting 100 to binary 0110 0100 and 10 to binary 0000 1010
- ▪
- Subtraction method - converting 10 to – 10 and adding // direct
- subtraction ...
- ▪
- ... correct answer 0101 1010
- Method 1: Converting to -10 and adding:
- Binary for +10 is 0000 1010
- Binary for –10 is 1111 0110
- Binary for 100 is 0110 0100

Solution 7

(c) Worked steps

Expand in base 16, or convert through binary — the mark scheme accepts either, and doing both is a free check.

- `C0F`: $12 \times 256 + 0 \times 16 + 15 \times 1 = 3072 + 0 + 15 = \mathbf{3087}$
- Through binary: `C0F` is `1100 0000 1111`, whose ones stand at $2048 + 1024 + 8 + 4 + 2 + 1 = 3087$. ✓

One mark for the working and one for the answer, so write the products out.
The middle digit is 0 and it still occupies its column. Dropping it makes the number `CF`, worth 207 — a mistake that is invisible once the working has been erased, which is why the place values are worth writing above the digits first. **Mark scheme**

Solution 7

- 1 mark for working:
- 1100 0000 1111 // 2048 + 1024 + 8 + 4 + 2 + 1
- // (12 * 162) + 15 // (12 * 16 * 16) + 15 // 3072 + 15
- 1 mark for correct answer:
- 3087
- 2
- 9618/12
- May/June 2024