Exercise walk-through

Number systems ▪ 1.1

eXercise

You must be able to

- convert between denary, binary and hexadecimal
- add binary numbers and identify overflow
- carry out a logical shift; use two's complement for negatives

Method — Conversions and two's complement

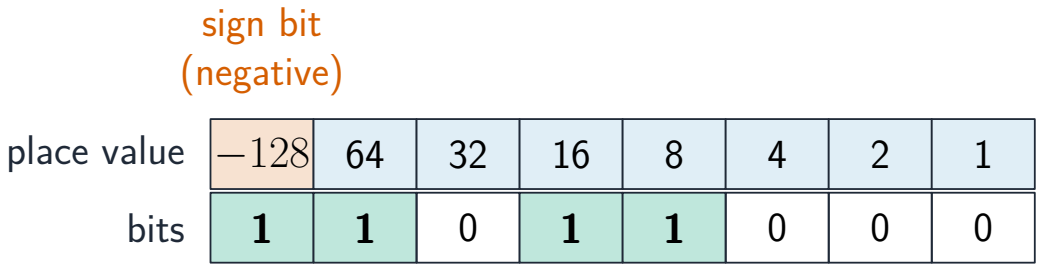
Denary 100 → binary: $64 + 32 + 4 \rightarrow$ 01100100 $\rightarrow$ hex 64.

place value	128	64	32	16	8	4	2	1
bits	**1**	0	0	**1**	0	**1**	**1**	0

$$150 = 128 + 16 + 4 + 2 \Rightarrow 10010110$$

An 8-bit place-value chart with 150 placed

Method — Conversions and two's complement



$$-128 + 64 + 16 + 8 = -40$$

A two's complement chart: MSB = −128

Method — The general methods, step by step

Denary to binary — divide by 2 and read the remainders bottom-to-top.

step	division	remainder
1	$150 \div 2 = 75$	0
2	$75 \div 2 = 37$	1
3	$37 \div 2 = 18$	1
4	$18 \div 2 = 9$	0
5	$9 \div 2 = 4$	1
6	$4 \div 2 = 2$	0
7	$2 \div 2 = 1$	0
8	$1 \div 2 = 0$	1

Method — The general methods, step by step

Reading bottom-to-top: `10010110`. Check with place values: $128 + 16 + 4 + 2 = 150$.
Binary to hexadecimal — split into nibbles of four bits from the right, then write each as one hex digit: `1100 1010` → `C A` → `CA`. Going back, expand each hex digit into four bits.
Binary addition. Add column by column from the right, carrying 1 when a column makes 2 (`1 + 1 = 10`). If a carry leaves the eighth column, the answer needs a ninth bit that the register cannot hold: that is **overflow** 溢出, and the stored answer is wrong.
Logical shift. A left shift of *n* places moves every bit *n* to the left, fills the right with zeros and multiplies the value by 2^n; a right shift divides by 2^n. Bits shifted out are lost, so a left shift can also lose the value.

Method — The general methods, step by step

Two's complement. Write the negative number by starting from the positive binary, inverting every bit, then adding 1. To read one, the leftmost bit is worth -128: $11011000 = -128 + 64 + 16 + 8 = -40$.

Why hexadecimal is used: it is shorter than binary and easier for a person to read and copy without mistakes, and each hex digit maps to exactly four bits. It appears in MAC addresses, IP version 6 addresses, HTML colour codes, memory dumps and error codes.

Practice 2.1 ▪ 2 marks

Convert the denary number 45 to 8-bit binary.

Solution 2.1

Worked steps

$45 = 32 + 8 + 4 + 1 \rightarrow$ `00101101` M1 A1.

Practice 2.2 ▪ 2 marks

Convert the binary number 11001010 to hexadecimal.

Solution 2.2

Method: The general methods, step by step

Worked steps

1100 1010 $\rightarrow$ C and A $\rightarrow$ CA M1 A1.

Practice 2.3 ▪ 3 marks

Convert the denary number 200 to 8-bit binary using the division method, showing your working.

Solution 2.3

Method: The general methods, step by step

Worked steps

$200 \div 2 = 100$ r0; $100 \div 2 = 50$ r0; $50 \div 2 = 25$ r0; $25 \div 2 = 12$ r1; $12 \div 2 = 6$ r0; $6 \div 2 = 3$ r0; $3 \div 2 = 1$ r1; $1 \div 2 = 0$ r1 M1 M1 *method, all remainders*; reading bottom-to-top, 11001000 A1.

Practice 2.4 ▪ 2 marks

Convert the hexadecimal number `3F` to denary.

Solution 2.4

Method: The general methods, step by step

Worked steps

$3 \times 16 + 15 = 48 + 15$ M1 $= 63$ A1.

Practice 2.5 ▪ 4 marks

Give two reasons why hexadecimal is used to represent binary numbers, and name two places where it is used.

Solution 2.5

Method: The general methods, step by step

Worked steps

Any two reasons B2: it is shorter than binary, so it takes less space to write; it is easier for a person to read, copy and remember without mistakes; each hex digit is exactly four bits, so converting is quick. Any two uses B2: MAC addresses; IP version 6 addresses; HTML colour codes; memory dumps; error codes; assembly language and debugging.

Exam-style 3.1 ▪ 1 mark

An 8-bit register can hold the denary values:

A 0 to 127

B 0 to 255

C 0 to 256

D 1 to 255

Solution 3.1

Method: The general methods, step by step

A 0 to 127

B 0 to 255 ✓

C 0 to 256

D 1 to 255

Worked steps

B. An 8-bit register holds 0 to 255.

Exam-style 3.2 ▪ 2 marks

Two 8-bit binary numbers are added: 01101100 + 00111010.
(a) Work out the binary result.
(b) Convert your answer to denary, and state whether overflow has occurred.

Solution 3.2

Method: The general methods, step by step

Worked steps

(a) 10100110 M1 A1.

(b) $128 + 32 + 4 + 2 = 166$; no overflow (the result fits in 8 bits) M1 A1.

Exam-style 3.3 ▪ 2 marks

A register holds `00110100`. Perform a logical left shift of 2 places and state the effect on the denary value.

Solution 3.3

Method: The general methods, step by step

Worked steps

11010000; the value is multiplied by 4 (a left shift of $2 = \times 2^2$) M1 A1.

Exam-style 3.4 ▪ 2 marks

A microprocessor uses 8-bit registers.
(a) Convert the denary number 172 to 8-bit binary.
(b) Convert your answer to (a) to hexadecimal.
(c) The register holds 10110101. This binary number is added to 01011100. Complete the addition and give the 8-bit answer.
(d) Explain whether overflow has occurred in part (c), and state what overflow means.
(e) The register 00011010 has a logical left shift of 3 places applied. Give the result, and calculate the effect on the denary value.
(f) The same register is used to hold two's complement numbers. Give the denary value of 11101100.

Solution 3.4

Method: The general methods, step by step

Worked steps

(a) $172 = 128 + 32 + 8 + 4$ **M1**, so 10101100 **A1**.
(b) 1010 1100 → A and C **M1**, so AC **A1**.
(c) 10110101 + 01011100 **M1** **M1** *carries shown correctly, addition to eight columns* = 00010001 with a carry out of the eighth column **A1**.
(d) Yes, overflow has occurred **B1**: the true answer needs nine bits, but the 8-bit register cannot store the ninth, so the stored answer is wrong **B1**.
(e) 11010000 **B1**; the denary value was 26 and becomes 208 **B1**; a left shift of 3

Solution 3.4

places multiplies by $2^3 = 8$, and $26 \times 8 = 208$ B1.
(f) $-128 + 64 + 32 + 8 + 4 = -20$ M1 A1.