IGCSE COMPUTER SCIENCE • EXERCISE SHEET

1.1 Number systems

Name: _____ Class: _____ Date: _____ **Total: 11 marks**

Objective

Build the skills to answer exam questions on **number systems** 数制—denary/binary/**hexadecimal** 十六进制 conversion, **binary addition** 二进制加法, **overflow** 溢出, shifts, and **two's complement** 补码.

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

1 Worked examples

■ Conversions and two's complement

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`

Place values 128⋯1; the 1s sit under the values that sum to the number

	sign bit (negative)							
place value	−128	64	32	16	8	4	2	1
bits	**1**	**1**	0	**1**	**1**	0	0	0

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

Two's complement: the MSB is worth −128, so 11011000 = −40

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

2 Practice

2.1 Convert the denary number 45 to 8-bit binary. [2]

2.2 Convert the binary number `11001010` to hexadecimal. [2]

3 Exam-style questions

3.1 An 8-bit register can hold the denary values: [1]

- **A** 0 to 127
- **B** 0 to 255
- **C** 0 to 256
- **D** 1 to 255

3.2 Two 8-bit binary numbers are added: `01101100` + `00111010`.

(a) Work out the binary result. [2]

(b) Convert your answer to denary, and state whether overflow has occurred. [2]

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

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **1.1 Number systems** past-paper sheet in the Library, or try these in **Practice mode**:

- 0478/12 N25 —Q1 (number systems)
- 0478/12 N25 —Q2 (binary / hex)

Solutions

2.1 $45 = 32 + 8 + 4 + 1 \rightarrow$ `00101101`.

2.2 `1100 1010` $\rightarrow$ `C` and `A` $\rightarrow$ `CA`.

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

3.2 (a) `10100110`.
(b) $128 + 32 + 4 + 2 = 166$; no overflow (the result fits in 8 bits).

3.3 `11010000`; the value is multiplied by 4 (a left shift of $2 = \times 2^2$).