

1.1 Number systems

Name: _____ Class: _____ Date: _____ **Total: 34 marks**

KEY WORDS denary 十进制 • hexadecimal 十六进制 • binary addition 二进制加法
• overflow 溢出 • two's complement 补码

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: $64 + 32 + 4 \rightarrow 01100100 \rightarrow$ hex 64.

■ 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

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: $1100\ 1010 \rightarrow \text{C A} \rightarrow \text{CA}$. 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.

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.

2 Practice

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

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

2.3 Convert the denary number 200 to 8-bit binary using the division method, showing your working. [3]

2.4 Convert the hexadecimal number 3F to denary. [2]

2.5 Give two reasons why hexadecimal is used to represent binary numbers, and name two places where it is used. [4]

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]

3.4 A microprocessor uses 8-bit registers.

(a) Convert the denary number 172 to 8-bit binary. [2]

(b) Convert your answer to (a) to hexadecimal. [2]

(c) The register holds 10110101. This binary number is added to 01011100. Complete the addition and give the 8-bit answer. [3]

(d) Explain whether overflow has occurred in part (c), and state what overflow means. [2]

(e) The register 00011010 has a logical left shift of 3 places applied. Give the result, and calculate the effect on the denary value. [3]

(f) The same register is used to hold two's complement numbers. Give the denary value of 11101100. [2]

Solutions

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

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

2.3 $200 \div 2 = 100$ 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; reading bottom-to-top, 11001000.

2.4 $3 \times 16 + 15 = 48 + 15 = 63$.

2.5 Any two reasons: 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: MAC addresses; IP version 6 addresses; HTML colour codes; memory dumps; error codes; assembly language and debugging.

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 $2 = \times 2^2$).

3.4 (a) $172 = 128 + 32 + 8 + 4$, so 10101100.

(b) 1010 1100 $\rightarrow$ A and C, so AC.

(c) $10110101 + 01011100 = 00010001$ with a carry out of the eighth column.

(d) Yes, overflow has occurred: the true answer needs nine bits, but the 8-bit register cannot store the ninth, so the stored answer is wrong.

(e) 11010000; the denary value was 26 and becomes 208; a left shift of 3 places multiplies by $2^3 = 8$, and $26 \times 8 = 208$.

(f) $-128 + 64 + 32 + 8 + 4 = -20$.