

3.1 Number representations and codes

Name: _____ Class: _____ Date: _____

Total: 23 marks

KEY WORDS binary • hexadecimal • two's complement
• overflow

Objective

Convert between binary, hexadecimal and decimal; negate in two's complement; detect overflow by sign reasoning – and verify every conversion by converting back.

1 Key knowledge and worked examples

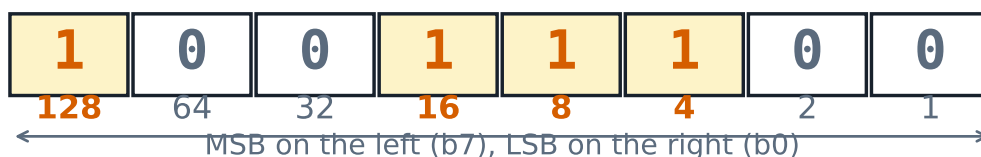
■ Positional conversion

Integer conversion is repeated division by the base, collecting remainders; hexadecimal groups binary in fours from the right. Every conversion checks by converting back or re-adding the weights.

Worked example. $156 = 10011100_2 = 9C_{16}$. Check by weights: $128 + 16 + 8 + 4 = 156 \checkmark$. Hex check: $9C = 9 \times 16 + 12 = 156 \checkmark$.

$$128 + 16 + 8 + 4 = 156$$

hex: $1001\ 1100 = 0x9C$



■ Two's complement: invert and add one

To negate an N-bit number: invert every bit, add 1. -5 in 8 bits: $00000101 \rightarrow 11111010 + 1 = 11111011$.

The **range** in N bits is $-2^{N-1} \dots 2^{N-1} - 1$; the extreme value $1000\dots 0$ is its own negative – negating it is **overflow**.

■ Overflow detection: like signs in, unlike sign out

Adding two same-sign numbers that yields the opposite sign is overflow – the result left the range. Adding unlike signs never overflows. **Worked example:** 0111 0000 (112) + 0000 1000 (8) = 0111 1000 (120) fine; but 0111 0000 + 0001 0000 (16) = 1000 0000 (–128 as signed) – two positives made a negative: overflow ✓ (the true sum 128 is out of the +127 range).

2 Practice

2.1 Convert 200 to binary and to hex; check by re-adding the weights. [3]

2.2 Write –46 in 8-bit two’s complement, and verify by adding it to +46. [3]

2.3 Add the 8-bit signed values 0101 1010 and 0100 0110. Give the result as signed decimal and state whether overflow occurred, with the reason. [4]

3 Exam-style questions

3.1 A sensor reports angles in 10-bit two's complement covering $-512^\circ \dots +511^\circ$.

State the binary for $+200^\circ$ and for -200° . Then add $+400^\circ$ and $+200^\circ$ as 10-bit values, give the signed result, and explain the overflow using the sign rule and the range. State the largest positive angle that can safely be doubled. [7]

3.2 An 8-bit system computes $X - Y$ as $X + (-Y)$ in two's complement.

Compute $90 - 30$ through the full binary working (convert, negate, add), verify the decimal result, and state why the subtraction method needs no special borrow logic. Then compute $90 - (-90)$ and explain any anomaly. [6]

Solutions

2.1

- $200 = 11001000_2$.
- Hex: C8.
- Check: $128 + 64 + 8 = 200 \checkmark$.

2.2

- $+46 = 0010\ 1110$.
- Invert $1101\ 0001$, add 1: $1101\ 0010$.
- Verify: $0010\ 1110 + 1101\ 0010 = (1)0000\ 0000$ – the carry-out discards, leaving zero $\checkmark$.

2.3

- $0101\ 1010\ (90) + 0100\ 0110\ (70)$: binary sum $1010\ 0000$.
- As signed: leading bit 1 $\rightarrow$ negative: $1010\ 0000 = -(0101\ 1111 + 1) = -96$.
- Overflow: yes – two positives produced a negative (true sum $160 > 127$).

3.1

- $+200 = 0011001000$ (10 bits).
- -200 : invert 1100110111 , add 1 $\rightarrow 1100111000$.
- $+400 = 0110010000$; adding $+200$: $0110010000 + 0011001000 = 1001011000$.
- Leading bit 1: signed value is negative ($-(0110100111 + 1) = -(0110101000) = -424$) – but the true sum is 600, out of range: overflow, sign rule confirmed (two positives made a negative).
- Largest safe double: $2X \leq 511 \rightarrow X = 255$ ($255+255 = 510 \checkmark$ fits).

3.2

- $90 = 0101\ 1010$; $30 = 0001\ 1110$; $-30 = 1110\ 0010$.
- $0101\ 1010 + 1110\ 0010 = (1)0011\ 1100 \rightarrow$ discard carry $\rightarrow 0011\ 1100 = 60 \checkmark$.
- No borrow logic because two's complement addition is modulo- 2^8 : subtraction becomes addition of the additive inverse, and the shared adder serves both.
- $90 - (-90) = 180$: $-(-90)$ requires negating $1010\ 0110$: $0101\ 1001 + 1 = 0101\ 1010 = 90$; adding gives $0101\ 1010 + 0101\ 1010 = 1011\ 0100$ – a negative result from two positives: overflow ($180 > 127$).