UNIVERSITY COMPUTER SCIENCE • EXERCISE SHEET

# 3.1 Binary and data encoding

Name: _____ Class: _____ Date: _____ **Total: 24 marks**

**KEY WORDS** two's complement • IEEE-754 IEEE-754

## Objective

Convert representations; catch overflow and float surprises.

## 1 Key knowledge and worked examples

### ■ The readings

Unsigned: 0 to $2^n - 1$. Two's complement: flip all bits, add 1 – $-b \equiv 2^n - b \pmod{2^n}$, so subtraction reuses the adder ✓. IEEE-754: sign, biased exponent, fraction.

### ■ Worked example: overflow read both ways

$96 + 64 = 160$ fits unsigned 8-bit but overflows signed (max 127) – the stored pattern $10100000_2$ reads $-96$ ✓. And $0.1 + 0.2 = 0.30000000000000004$: rounding, not a bug.

## 2 Practice

**2.1** Convert 45 to 8-bit binary. [3]

**2.2** Write $-50$ as 8-bit two's complement. [3]

**2.3** The character A has ASCII code 65 – what stores the character 1? [3]

## 3 Exam-style questions

**3.1** All arithmetic in 8-bit two's complement.

Compute (i) $60 + 50$, (ii) $-60 - 50$, (iii) $100 + 55$, stating for each whether overflow occurs and what the stored pattern reads. Explain in one line why unsigned and signed overflow look different in the same adder. [8]

**3.2** IEEE-754 single precision has 1 sign, 8 exponent, 23 fraction bits with bias 127.

State the largest finite float shape (exponent all-ones excluded) and why 0.1 cannot be stored exactly. Explain why float equality tests after arithmetic are unsafe, and what to do instead. [7]

## Solutions

**2.1**

- $45 = 32 + 8 + 4 + 1 = 00101101_2$ ✓.

**2.2**

- $50 = 00110010_2$; flip 11001101; add $1 \to 11001110$ ✓.

**2.3**

- The code 49 – a code, not a value ✓.

**3.1**

- (i) 110, in range ✓.
- (ii) $-110$, in range $(-128..127)$ ✓.
- (iii) $155 > 127$: overflow; pattern reads $155 - 256 = -101$ ✓.
- Unsigned overflow is carry out of bit 7; signed overflow is when the carry INTO the sign bit differs from the carry out – same adder, two tests ✓.

**3.2**

- Largest: exponent 254 (biased) $\to 2^{127} \times \sim 1.111\ldots_2 \approx 3.4 \times 10^{38}$ ✓.
- $0.1 = 0.000110011\ldots$ repeating – no finite binary form, so it rounds ✓.
- Equality is unsafe because two differently-rounded paths reach slightly different values; compare $|a - b| < \varepsilon$ instead ✓.

exercises.lol exercises.com.cn