NAME	DATE STARTED	TARGET END DATE

GCE AS LEVEL — COMPUTER SCIENCE UNIT 1 QUESTION PACK

2500U10-1 · 2015 spec Unit 1 Topic 3 · AS unit, first sat 2017, 100 marks, 2h paper

REVISE.wales

COMPUTER SCIENCE — UNIT 1 · Data Representation — Binary, Hex & Two's Complement

Topic 1.2 — Number systems, binary arithmetic, two's complement and floating point

Primitive data types and storage requirements, denary↔binary↔hex conversions, binary addition, two's complement representation of signed integers, floating point form (mantissa and exponent), and word length.

2015 specification · current

Estimated time for entire question pack: ~2 h 12 min

Derived from the Unit 1 pace of ~1.2 min/mark, padded for written-prose answers (88 marks over 8 questions).

*You are advised to **not** attempt to complete all of this in one sitting.*

ABOUT THIS QUESTION PACK

This is a **comprehensive topic question pack**, not a single mock paper. It contains every question from the WJEC AS Unit 1 papers (Summer 2017 – Summer 2024, COVID gap) that maps onto Topic 1.2 of the 2015 specification.

Questions are ordered by source paper date.

INSTRUCTIONS

Use black ink or black ball-point pen. Show all working. A calculator is allowed where useful.

For Examiner's use only

Q	Source	Max	Mark
1	S17 Q4	8	
2	S17 Q7	11	
3	S18 Q5	9	
4	S19 Q11	13	

Q	Source	Max	Mark
5	S22 Q4	19	
6	S23 Q1	8	
7	S23 Q14	12	
8	S24 Q4	8	
Total		88	

Data Representation — Binary, Hex & Two's Complement — what the spec asks

WJEC GCE AS Computer Science (from 2015) · Unit 1: Fundamentals of Computer Science · Topic 1.2.

Number bases

- Denary (base 10), binary (base 2), hex (base 16) using digits 0–9 and A–F.
- Hex digit = 4 binary bits; pairs of hex = 1 byte (8 bits).
- Place values: binary 128 64 32 16 8 4 2 1; hex 16^0, 16^1, 16^2...
- Convert denary → binary by repeated division-by-2; binary → hex by grouping 4-bit nibbles.

Binary addition rules

- 0 + 0 = 0; 0 + 1 = 1; 1 + 1 = 10 (carry 1); 1 + 1 + 1 = 11 (carry 1).
- Add right-to-left, propagating carries.
- Overflow occurs when carry-out of MSB cannot be stored in fixed-width result.
- Show working clearly — method marks are awarded even if final answer is wrong.

Two's complement

- Signed n-bit range: -2^{n-1} to $(2^{n-1} - 1)$. 8-bit: −128 to +127.
- Positive numbers stored normally; MSB = 0 indicates positive.
- Negate: flip all bits then add 1. MSB = 1 indicates negative.
- Subtraction A − B = A + (−B) using two's complement of B.

Floating point

- Form: mantissa × $2^{exponent}$. Mantissa stores significant digits, exponent the scale.
- Normalised form: leading bit after point is 1 (or 1 before point) for max precision.
- Larger exponent ⇒ wider range; larger mantissa ⇒ higher precision.
- Trade-off: same total bits, choose between range and precision.

Character encoding

- ASCII: 7-bit, 128 characters (digits, letters, basic punctuation).
- Extended ASCII: 8-bit, 256 characters incl. accented letters.
- Unicode (UTF-8, UTF-16): supports all languages; variable-length in UTF-8.
- Storage: 1 byte per ASCII char vs 2–4 bytes per Unicode char.

Word & storage

- Word length = number of bits CPU handles in one operation (32, 64, …).
- Larger word ⇒ more memory addressable + faster integer arithmetic.
- Primitive types: integer, real, char, Boolean — choose minimum bits for given range.
- 1 byte = 8 bits; 1 kibibyte = 1024 bytes; 1 mebibyte = 1024 KiB.

Data Representation — Binary, Hex & Two's Complement in one page

Quick-reference notes — revisit before each question.

Quick binary ⇔ hex

Group binary into 4-bit nibbles from the right.
0000=0 0001=1 … 1001=9 1010=A 1011=B 1100=C 1101=D 1110=E 1111=F
Reverse: write each hex digit as its 4-bit binary.

Two's complement negate

Flip all bits then add 1.
e.g. 5 (8-bit) = 00000101
flip → 11111010, +1 → 11111011 = −5.

Binary addition carries

0+0=0, 0+1=1, 1+1=10 (carry 1), 1+1+1=11 (carry 1).
Add column-by-column right to left, propagate carries.
Show each column working.

Overflow detection

For signed two's complement: overflow if the carries into and out of the MSB differ.
Or: adding two positives gives negative, or two negatives gives positive.

Floating point form

value = mantissa × $2^{exponent}$.
Normalise: mantissa starts 0.1xxx (positive) or 1.0xxx (negative) so MSB carries information.
Big exponent ⇒ range; big mantissa ⇒ precision.

Character storage

ASCII 'A' = 65 (0x41).
'a' = 97 (0x61). '0' = 48.
Unicode covers >140,000 characters;
UTF-8 is variable length 1–4 bytes.

Examiner only

4. Different primitive data types are used in computer systems.

(a) (i) Using the denary example 108_{10}, calculate the minimum storage requirements for an integer data type within a range of 0_{10} to 127_{10}. [2]

....

....

....

....

....

....

(ii) In a certain computer system, numbers are represented using sign and magnitude.

Give the range for a **signed** integer data type with the same storage requirements as question 4*(a)*(i). [1]

....

....

....

....

(b) Describe the use and advantages of the Unicode standardised character set. [3]

....

....

....

....

....

....

(c) Giving suitable examples compare the storage requirements for a character and a string data type. [2]

....

....

....

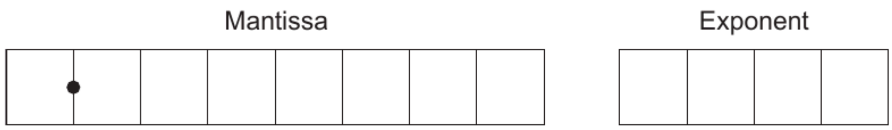
....

7. *(a)* Convert the denary numbers 87_{10} and 113_{10} into their equivalent unsigned 8 bit binary numbers.

Carry out the binary addition of the two resulting 8 bit binary numbers. Convert your binary answer into a hexadecimal number.

Show **all** of your workings. [5]

(b) (i) In a certain computer system, real numbers are stored in floating-point form using two's complementation, an 8 bit mantissa and a 4 bit exponent.



Convert the number 4.125_{10} into this floating-point form. [3]

Examiner only

(ii) In the same computer system, the following is a floating-point representation of a real number:

Mantissa									Exponent			
0	• 1	1	1	1	0	0	0		0	1	1	1

Calculate the denary value of the mantissa and exponent, and convert this floating-point number into a denary number. [3]

2500U101 09

Examiner only

5. (a) Convert 31_{16} and $6D_{16}$ into binary numbers and add them together using binary addition. [3]

(b) (i) In a certain computer system, real numbers are stored in floating point form using two's complementation, a 12 bit mantissa and a 4 bit exponent.

Convert the number 16.125_{10} into this floating point form. [3]

(ii) In a different computer system, real numbers are stored in floating point form using two's complementation, a 5 bit mantissa and a 3 bit exponent.

Showing your workings, calculate the largest positive denary number that this computer system can store. [3]

Examiner only

(c) Describe an appropriate circumstance for the use of each search type. [2]

....

....

....

....

....

....

11. (a) Convert $6C_{16}$ and AF_{16} into binary and add them together using binary addition. [3]

....

....

....

....

....

....

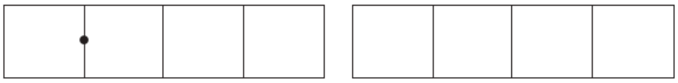
....

....

Examiner only

(b) In a certain computer system, real numbers are stored in **normalised** floating-point form using a positive 4 bit mantissa and a positive 4 bit exponent.

Mantissa Exponent



Calculate the denary range of positive real numbers that can be stored in this normalised floating-point form.

Show all your workings. [6]

Examiner only

4. (a) State the meaning of the term “word”. [1]

(b) Convert the hexadecimal numbers AE_{16} and $1B_{16}$ into binary and, using binary addition, calculate the number that would result from adding them.
Convert your answer into denary.

You must show **all** of your working. [5]

(c) Show how -39_{10} would be represented using sign/magnitude and how it would be represented using two’s complement representation in an 8 bit register. [2]

(d) In a certain computer system, real numbers are stored in normalised floating point form using two's complementation, an 8 bit mantissa and a 3 bit exponent.

(i) The following is a floating point representation of a real number:

0.1110100 010

Calculate the denary value of the mantissa and exponent, and convert this floating point number into a denary number. [3]

(ii) Showing your workings, calculate the largest positive denary number that this computer system can store. [3]

Examiner only

(e) Using the number 111.10110_2, describe truncation and rounding **to two binary places**, and their effect upon accuracy in terms of their absolute errors. State which method is more accurate. [5]

Examiner only

Answer **all** questions.

1. (a) Complete the table. [6]

Data Type	Example Data	Storage Requirements (in bits)
Character (ASCII)		b
Boolean		b
Short Integer		b

(b) Describe how character sets are used in a computer system. [2]

....

....

....

....

....

....

Examiner only

14. Andrew is considering using three different methods for storing **positive** numbers, using two's complementation, in a computer system.

METHOD 1	**METHOD 2**	**METHOD 3**
Store numbers as an integer using an 8 bit register.	Store real numbers in normalised floating-point form, using a 5 bit mantissa and a 3 bit exponent.	Store real numbers in normalised floating-point form, using a 4 bit mantissa and a 4 bit exponent.

Calculate the denary range (highest and lowest) of numbers that can be stored by each method. Compare the advantages of representing numbers in integer and floating-point forms. [12]

Examiner only

END OF PAPER

Examiner only

4. (a) Describe the term 'word length' in relation to a CPU. [1]

(b) Convert the denary number 27_{10} and the hexadecimal number $7E_{16}$ into two 8-bit binary numbers. Add the two numbers together using binary addition, leaving your answer in binary.

You must show all of your workings. [4]

(c) Using the binary number 101.0101_2, demonstrate truncation to one binary place, and calculate the effect upon accuracy in terms of relative error. [3]

END OF QUESTION PACK

8 questions · 88 marks · ~2 h 12 min
Source: WJEC AS Computer Science Unit 1 (2500U10-1), Summer 2017–2024, COVID gap
Curated for WJEC Computer Science 2015 spec AS Unit 1 — Topic 3 (1.2)