OXFORD AQA
INTERNATIONAL QUALIFICATIONS

INTERNATIONAL AS

COMPUTER SCIENCE

CS02

Unit 2 Concepts and principles of computer science

Mark scheme

Specimen

Version: 1.0 Final

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from oxfordaqaexams.org.uk

How to mark

Aims

When you are marking your allocation of scripts your main aims should be to:

- recognise and identify the achievements of students
- where relevant, place students in the appropriate mark band and in the appropriate part of that mark band (high, low, middle) for **each** assessment objective
- record your judgements with brief notes, annotations and comments that are relevant to the mark scheme and make it clear to other associates how you have arrived at the numerical mark awarded for each assessment objective
- ensure comparability of assessment for all students, regardless of question or examiner.

Approach

It is important to be **open minded** and **positive** when marking scripts.

The specification recognises the variety of experiences and knowledge that students will have. It encourages them to study computer science in a way that is relevant to them. The questions have been designed to give them opportunities to discuss what they have found out about computer science. It is important to assess the quality of **what the student offers**.

Do not mark scripts based on the answer **you** would have written. The mark schemes have been composed to assess **quality of response** and not to identify expected items of knowledge.

Assessment Objectives

This component requires students to:

AO1: Demonstrate knowledge and understand of the key concepts and principles of computer science.

AO2: Apply knowledge and understanding of key concepts and principles of computer science.

AO3: Analyse problems in computational terms in order to develop and test programmed solutions and demonstrate an understanding of programming concepts.

The following annotation is used in the mark scheme.

;	means a single mark
//	means alternative response
/	means an alternative word or sub-phrase
A	means acceptable creditworthy answer
R	means reject answer as not creditworthy
NE	means not enough
I	means ignore
DPT	in some questions a specific error made by a student, if repeated, could result in the student failing to achieve multiple marks. The **DPT** label indicates that this mistake should result in a student not achieving only one mark, on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated

Question	Part	Marking guidance	Total marks
01	1	**E** XOR; **R.** if more than one lozenge shaded	1 **A01 = 1**

Question	Part	Marking guidance	Total marks
01	2	The key must be (at least) as long as the data to be encrypted; The key must not be reused; The key must be (truly) random; The key must be kept securely/not revealed; **Max 2**	2 **AO1 = 2**

Question	Part	Marking guidance	Total marks
01	3	Symmetric: The same key is used to encrypt and decrypt; **A.** Sender and receiver use same key. Asymmetric: Different (but related) keys are for encryption and decryption; **A.** Sender and receiver use different keys. **NE.** Symmetric uses one key // asymmetric uses two keys. **Max 1**	1 **AO1 = 1**

Question	Part	Marking guidance	Total marks
02	1	(see level table below)	4 **AO1 = 4**

Level	Description	Marks
4	Description covers all, or almost all, of the points in the indicative guidance and fully reflects the sequence in which steps occur. It includes use of registers, buses and main memory. An excellent level of understanding is shown with no misconceptions.	4

3	Description covers most (ie more than half) of the points in the indicative guidance and completely or almost completely reflects the correct sequence in which steps occur. At least two of the use of registers, buses and main memory are covered. A good level of understanding is shown. Whilst there may be some omissions, there is at most one misconception in the response.	3
2	At least two correct points are made from the indicative guidance and there is some indication of understanding of the correct sequence. Some understanding is shown.	2
1	At least one relevant point has been made. There is not sufficient evidence to conclude that the cycle has been understood.	1
	The response does not meet the standard required for Level 1.	0

Guidance – Indicative Response

- Contents of Program Counter/PC transferred to Memory Address Register/MAR.
- Address bus used to transfer this address to main memory.
- Memory read signal sent to main memory using control bus.
- Fetched value/instruction transferred using the data bus.
- Contents of addressed memory location loaded into the Memory Buffer Register/MBR.
- Transfer content of Memory Buffer Register/MBR to the Current Instruction Register/CIR.

A. Memory Data Register/MDR for MBR
I. Incrementing of program counter, even if incorrect
NE. Points made using register transfer notation only eg CIR ← [MBR]

Question	Part	Marking guidance	Total marks
02	2	Instruction and data can be accessed simultaneously; Avoid/reduce bottleneck of single data/address bus(es) // avoid/reduce delays waiting for memory fetches; Avoids possibility of data being executed as code (which is one method that can be exploited by hackers); Being able to use exclusively ROM for instruction memory prevents the program being modified/hacked; **A.** program cannot be accidentally overwritten (by data); Instruction and data memory can have different word lengths; Different technologies can be used to implement instruction and data memory; Different quantities of instruction and data memory means that address lengths can differ between the two // memory address structures can differ; **Max 2** **NE.** So programs/tasks will run faster **NE.** More efficient	**2** **AO1 = 2**

Question	Part	Marking guidance	Total marks
03	1	Award **3 marks** for correct answer: 16.48 **A.** responses written correctly to more decimal places (16.4794921875) or as a fraction 16 491/1024 **A.** 48000 × 16 × 3 × 60 / 8 / 1024 / 1024 Award **2 marks** for an answer written to 0 or 1 decimal places (16 or 16.5) or if truncated to 16.47 If answer is incorrect then award **up to 2 method marks**. Award 1 method mark for every two steps completed from this list: • multiplying by 48000 • multiplying by 16 • multiplying by 3 • multiplying by 60 • dividing by 8 • dividing by 1024 / 2^{10} • dividing by 1024 / 2^{10} a second time	**3** **AO2 = 3**

		The following method points are equivalent to performing two of the method points in the list above: • multiplying by 180 • dividing by 2 • dividing by 1048576 / 2^{20} **Max 2** if answer is not correct and written to at least 2 decimal places	

Question	Part	Marking guidance	Total marks
03	**2**	Award **1 mark** for correct answer: 30000 **A.** 15000 × 2, double 15000	**1** **AO2 = 1**

Question	Part	Marking guidance	Total marks
03	**3**	Digital to Analogue Converter **A.** DAC **NE.** Digital to Analogue	**1** **AO1 = 1**

Question	Part	Marking guidance	Total marks
04	**1**	**1 mark**: Introduced to support a larger range of characters **1 mark** for any point from the list below: • Increased international communication. • Use of files in multiple countries. • Requirement to use additional symbols (allow examples, eg mathematical / scientific / engineering / emoji symbols). • Facilitates interchange of documents between countries. • Culturally unacceptable to only allow non-English speaking countries to communicate in English // (concurrent) support for multiple languages. **Max 2**	**2** **AO1 = 2**

Question	Part	Marking guidance	Total marks
04	2	**Alternative 1** The number of 1s is counted // the (1) bits are summed; if the count/sum/total is even, the parity bit is set to 0, otherwise it is set to 1 // if the count//sum/total is odd, the parity bit is set to 1, otherwise it is set to 0 // the parity bit is set to a value which ensures the total number of 1s is even; **Alternative 2** The bits are XOR'd with each other; and the result is the parity bit; **Max 2**	**2** **AO1 = 2**

Question	Part	Marking guidance	Total marks
04	3	0;	**1** **AO2 = 1**

Question	Part	Marking guidance	Total marks
04	4	Errors that change an even number of bits (A. two bits) cannot be detected; **R.** multi-bit errors cannot be identified (Errors can be detected but) errors cannot be corrected; **A.** Position of errors cannot be identified **Max 1**	**1** **AO1 = 1**

Question	Part	Marking guidance	Total marks
05	1	Software used in the management of a computer system // layer(s) of software that abstract the user from how the computer works // software that provides a platform for other software to use; **A.** software used to run the computer **A.** software that provides a virtual machine **NE.** software that maintains a computer	1 **AO1 = 1**

Question	Part	Marking guidance	Total marks
05	2	**B** Bitmap image editors; **R.** if more than one lozenge shaded	1 **AO1 = 1**

Question	Part	Marking guidance	Total marks
05	3	To allocate processors/cores to processes // schedule processes // decide which process to carry out when // manage the execution of multiple processes; **NE.** processor management To allocate memory/RAM to processes // to determine what areas of memory are used for what purpose // moving data into and out of RAM / to a paging file for virtual memory // ensuring processes can only write to memory that they have been allocated; **NE.** memory management To allocate I/O devices to processes // manages communication between processes and I/O devices // automatic installation of drivers for new I/O devices; **A.** examples of devices (but no more than one mark) **NE.** manages I/O devices To hide the complexities of the hardware from the user; **NE**. virtual machine without description **R.** user interface To handle interrupts // to call appropriate interrupt handler (**A.** ISR) when an interrupt occurs; To allocate space on a storage device to files // organising files into directories // determines where on a device to save a file // recognising storage devices when they are connected;	2 **AO1 = 2**

		A. defragmentation of disks **NE.** saving a file Installation of new software // automatic / managing updating of software; **A.** "programs" or "tasks" for "processes" Manage power consumption / use of battery; **A.** examples of this eg controlling clock speed, brightness of screen **Note:** Students must describe – phrases such as "processor management", "allocating memory" etc are not enough. **Max 2**	

Question	Part	Marking guidance	Total marks
06	**1**	Award **3 marks** if correct final answer is shown: 16 // 2^4 If final answer is not given then award **3 marks** if correct calculation is shown: $$2^{\left(\frac{400\times1000\times8}{1000\times800}\right)}$$ If final answer is not correct or overall calculation is not clear then award **up to 2 marks for working**, one for each of the points listed below: • multiplying 400 by 1000 // 400 000 shown in working; • dividing 3 200 000 or 400 000 or 8000 or 3200 or 400 or 8 by a number; • multiplying 1000 by 800 // 800 000 shown in working; • multiplying by 8 to convert from bytes to bits; • colour depth calculated as 4; • showing 2^x as the last stage of the working, where x is the value calculated so far;	**3** **AO2 = 3**

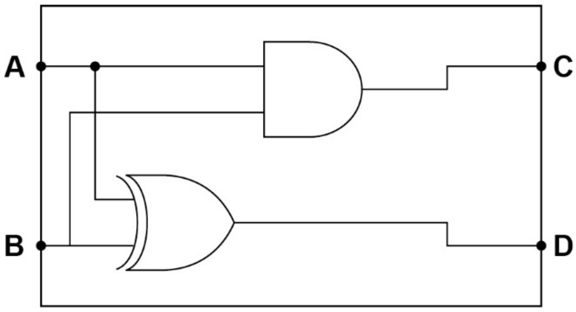
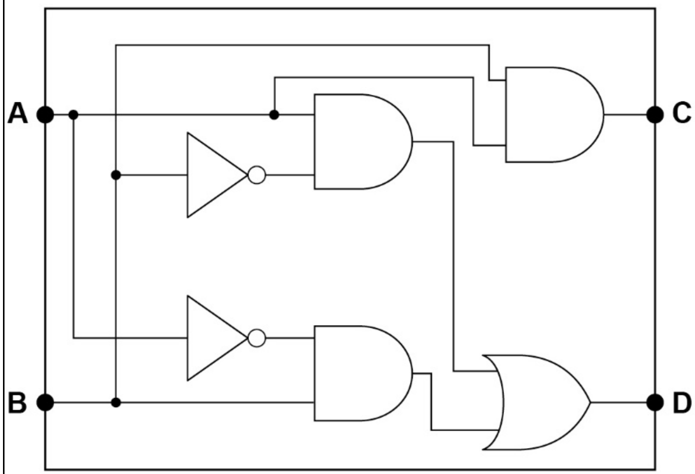
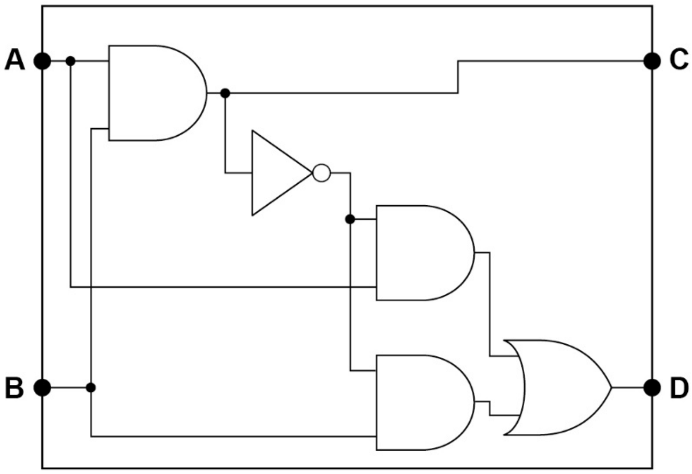
Question	Part	Marking guidance	Total marks
06	2	Bitmap images store the colour of each pixel // vector graphics do not need to store the colour of each pixel; **A.** data about pixel instead of colour, but **R.** just storing pixels The image contains 800 000 pixels // images can contain lots of pixels; Vector graphics store information about / properties of the objects that an image is composed of; **A.** “shapes” for “object” **R.** “equations” for “object” **R.** “instructions” for “object” unless clear that instructions are descriptions of objects **A.** examples of properties/information instead of the actual words, if there are at least two valid examples **NE.** vector graphics are composed of objects without reference to properties/information It takes only a small amount of memory to store the properties of an object; (Large) images can be composed of relatively few objects // there will be fewer objects than there would be pixels // a single object might be equivalent to many pixels; **Max 3**	**3** **AO1 = 3**

Question	Part	Marking guidance	Total marks
06	3	Individual objects / components / parts of the image can be manipulated / edited / duplicated / copied independently; **NE.** images are easy to edit The image / individual objects / components / parts of the image can be enlarged / scaled without loss of quality / without becoming pixelated // vector graphics are resolution independent; **A.** zoomed in” for enlarged **NE.** easy to scale If an object / component is deleted, the software knows what is behind it // no “hole” is left in the image;	**2** **AO1 = 2**

		R. faster transmission times (as a direct consequence of fewer bytes, given in question) **Max 2**	

Question	Part	Marking guidance	Total marks
07		**What cache memory is (Max 1):** * Memory that can be accessed very quickly; Memory located on (**A.** close to) the processor; **What cache memory is used for:** To store most frequently used // most recently used // pre-fetched instructions/data // to store instructions in the locality of the instruction currently being executed; **How more cache memory improves performance (Max 2):** More instructions/data can be stored in the cache; #Instructions/data stored in cache can be accessed more quickly <u>than instructions/data in main memory</u> // if an instruction is accessed <u>a second time</u>, it can be retrieved more quickly; This increases the probability that a particular data item/instruction is in the cache when fetched // this increases the probability of a cache hit // fewer fetches from main memory will be required; **Note:** Only award the point marked # if the point marked * has not already been awarded.	**4** **AO1 = 4**

Question	Part	Marking guidance	Total marks
08	1	**1 mark:** Output C is correct for all inputs **1 mark:** Output D is correct for all inputs **1 mark:** Circuit is fully correct and uses exactly two gates	**3** **AO2 = 3**

		Example circuits **3 mark solution:**  **2 mark solution:**  **2 mark solution:**  	

Question	Part	Marking guidance	Total marks
08	2	It adds two bits (**A.** numbers) together // it is a half adder; **A.** it is an adder as BOD, it performs addition **R.** it is a full adder	1 AO2 = 1

Question	Part	Marking guidance	Total marks
08	3	**Marking guidance for examiners** • Award marks for working out until an incorrect step has been made. • Ignore missing steps from the example solutions, as long as the jumps between steps are logically correct. • If, in any one step, a candidate is simplifying different parts of an expression simultaneously and makes an error, award marks for the correctly simplified part(s) and then stop marking. **1 mark** for final answer: A **3 marks** for working. Award up to **three marks** for applying each one of the three techniques (**one mark** per application, multiple marks can be awarded for using the same technique more than once): • a successful application of De Morgan's Law (and any associated cancellation of NOTs) that produces a simpler expression – award 2 marks if De Morgan's Law applied twice simultaneously • applying an identity other than cancelling NOTs that produces a simpler expression • successfully expanding brackets // factorising. **Note:** A simpler expression is one that is logically equivalent to the original expression but uses fewer logical operators. **Max 2** for working if there is no successful application of De Morgan. **Max 3** overall if any incorrect working	4 AO2 = 4

Example solution 1

$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot (\overline{\overline{\mathrm{A}} + (\overline{\mathrm{B}} \cdot \mathrm{C})})$	
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\overline{\mathrm{B}} \cdot \mathrm{C}}$	Application of De Morgan
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} \cdot \overline{\overline{\mathrm{B}} \cdot \mathrm{C}})$	Factorising
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} \cdot (\mathrm{B} + \overline{\mathrm{C}}))$	Application of De Morgan
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{B} + \mathrm{B} \cdot \overline{\mathrm{C}})$	Expand brackets
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} + \mathrm{B} \cdot \overline{\mathrm{C}})$	By $X \cdot X = X$
$\mathrm{A} \cdot (1 + \mathrm{B} \cdot \overline{\mathrm{C}})$	By $X + \overline{X} = 1$
$\mathrm{A} \cdot 1$	By $X + 1 = 1$
A	By $X \cdot 1 = X$

Example solution 2

$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot (\overline{\overline{\mathrm{A}} + (\overline{\mathrm{B}} \cdot \mathrm{C})})$	
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot (\overline{\overline{\mathrm{A}} + (\overline{\mathrm{B} + \overline{\mathrm{C}}})})$	Application of De Morgan
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{A} \cdot (\mathrm{B} + \overline{\mathrm{C}})$	Application of De Morgan
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{A} \cdot \mathrm{B} + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\mathrm{C}}$	Expand brackets
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{A} \cdot \mathrm{B} + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\mathrm{C}}$	By $X \cdot X = X$
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B}) + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\mathrm{C}}$	Factorising partially
$\mathrm{A} \cdot 1 + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\mathrm{C}}$	By $X + \overline{X} = 1$
$\mathrm{A} + \mathrm{B} \cdot \mathrm{A} \cdot \overline{\mathrm{C}}$	By $X \cdot 1 = X$
A	By $X + (X \cdot Y) = X$

Example solution 3

$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot (\overline{\overline{\mathrm{A}} + (\overline{\mathrm{B}} \cdot \mathrm{C})})$	
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot (\overline{\overline{\mathrm{A}} + (\overline{\mathrm{B} + \overline{\mathrm{C}}})})$	Application of De Morgan
$\mathrm{A} \cdot \overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{A} \cdot (\mathrm{B} + \overline{\mathrm{C}})$	Application of De Morgan
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} \cdot (\mathrm{B} + \overline{\mathrm{C}}))$	Factorising
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} \cdot \mathrm{B} + \mathrm{B} \cdot \overline{\mathrm{C}})$	Expanding brackets
$\mathrm{A} \cdot (\overline{\mathrm{B}} + \mathrm{B} + \mathrm{B} \cdot \overline{\mathrm{C}})$	By $X \cdot X = X$
$\mathrm{A} \cdot (1 + \mathrm{B} \cdot \overline{\mathrm{C}})$	By $X + \overline{X} = 1$
$\mathrm{A} \cdot 1$	By $X + 1 = 1$
A	By $X \cdot 1 = X$

		Example solution 4 $A \cdot \overline{B} + B \cdot (\overline{\overline{A} + (\overline{B} \cdot C)})$ $A \cdot \overline{B} + B \cdot (\overline{\overline{A} + (\overline{B + \overline{C}})})$ Application of De Morgan $A \cdot \overline{B} + B \cdot A \cdot (B + \overline{C})$ Application of De Morgan $A \cdot \overline{B} + B \cdot A \cdot B + B \cdot A \cdot \overline{C}$ Expand brackets $A \cdot \overline{B} + B \cdot A + B \cdot A \cdot \overline{C}$ By $X \cdot X = X$ $A \cdot \overline{B} + B \cdot A$ By $X + (X \cdot Y) = X$ $A \cdot (\overline{B} + B)$ Factorising $A \cdot 1$ By $X + \overline{X} = 1$ A By $X \cdot 1 = X$	

Question	Part	Marking guidance	Total marks
09	1	**Advantages of high-level language (MAX 2):** Program code is easier to understand/maintain/debug; Faster development time // programmers can be more productive // one line of HLL code can do the same job as many lines of assembly language; Programs are (more) portable (to other hardware platforms). Availability of flow control structures; **A.** Example(s) eg loops, selection Improved features for supporting modularity; **A.** Ability to use subroutines Built-in support for data structures; **A.** Example(s) eg arrays, records Language is problem-oriented; Support for different paradigms; **A.** Examples eg functional programming **Disadvantages of high-level language (MAX 2):** Assembly language code may execute more quickly; **R.** If response suggests that faster execution is because translation is not required Assembly language code may use less memory; Assembly language gives direct/better access to computer hardware // Enables direct manipulation of memory (contents); Reliance on the correctness of the compiler/interpreter; **NE.** “More efficient” for either executes more quickly or uses less memory Award marks for disadvantages as opposite of advantage points eg a disadvantage of assembly language could be “Program code is harder to understand/maintain”. **BUT** do not award two marks for an advantage and its corresponding disadvantage.	4 AO1 = 4

Question	Part	Marking guidance	Total marks
09	2	Software must be used to finish the translation process (on the computer running the program); The (JIT compiler) compiles the intermediate language code / bytecode into machine code for the processor / platform / computer it is being executed on; Use a virtual machine // a just-in-time / JIT compiler; (The virtual machine will) interpret / translate / execute the intermediate language code / bytecode // call functions within its own code to carry out the command; Each processor instruction set / architecture will have its own virtual machine; **Max 2**	**2** **AO1 = 2**

Question	Part	Marking guidance	Total marks
09	3	Allows processor / platform independence // code is more portable / cross-platform; Compiler is only required to translate source code once (but the intermediate language / bytecode can still be executed on a variety of platforms); Can create intermediate language code / bytecode for a specific (standardised) virtual machine that all the target machines have installed; The virtual machine can perform security checks on the (intermediate language / byte) code without execution, hence it is more secure; **Max 1**	**1** **AO1 = 1**

Question	Part	Marking guidance	Total marks
10	1	C;	**1** **AO2 = 1**

Question	Part	Marking guidance	Total marks
10	2	B;	1 AO2 = 1

Question	Part	Marking guidance	Total marks
10	3	Mantissa: 1 ● 0 1 1 0 0 0 0 Exponent: 0 0 1 1 Answer = -5 **If answer is correct award 2 marks.** **If answer is incorrect award 1 method mark** for one of: • showing correct value of both mantissa and exponent in denary (mantissa = -0.625 // -5/8, Exponent = 3) • showing binary point shifted 3 places to right in binary number • indicating that final answer calculated using answer = mantissa x $2^{exponent}$	2 AO2 = 2

Question	Part	Marking guidance	Total marks
10	4	Answer: Mantissa: 0 ● 1 1 1 0 1 0 1 Exponent: 0 1 1 0 **If answer is correct award 3 marks.** **If answer is incorrect award up to 2 method marks:** • Correct representation of 58.5 in fixed point binary: 111010.1; **A.** leading 0s.	3 AO2 = 3

		• Showing the correct value of the exponent in denary (6) or binary (110) // showing the binary point being shifted 6 places; **MAX 2**	

Question	Part	Marking guidance	Total marks
10	**5**	0.05 // 13.8 – 13.75; **R.** -0.05 **A.** Award BOD mark if correct method has been shown ie 13.8 – 13.75 but candidate has then made an error performing the subtraction operation	**1** **AO2 = 1**

Question	Part	Marking guidance	Total marks
10	**6**	0.36%; **A.** 0.0036 // 0.05 ÷ 13.8 **A.** Follow-through of incorrect answer to question part 11.5 **A.** Award BOD mark if correct method has been shown but candidate has then made an error performing the division operation.	**1** **AO2 = 1**

Question	Part	Marking guidance	Total marks
11	**1**	To store data / programs whilst the computer is turned off; **A.** long-term / permanent storage **NE.** secondary storage devices are non-volatile **NE**. store data that is not in use (As) the contents of RAM are lost when the computer is turned off; **R.** “main memory” for “RAM” **A.** main memory (RAM) To transfer data / programs between computers; **NE.** secondary storage devices are portable Allows the storage of data sets / files that could not fit in RAM // computer architecture supports a limited amount of main memory/RAM; **A.** primary store for main memory **NE.** to extend storage capacity, to store more, to store large files, higher capacity **A.** To use as virtual memory **Max 2**	**2** **AO1 = 2**

Question	Part	Marking guidance	Total marks
11	2	**All marks AO1 (understanding)** **Level** | **Description** | **Mark Range** 2 | A good understanding of the operation of an SSD has been demonstrated. The response is well structured, covers most of the points in the indicative content and does not contain any errors of understanding. | 3-4 1 | Some relevant points have been made, but the description omits important details or contains some errors so that only a limited understanding is demonstrated. | 1-2 Indicative content: • Data is stored electronically // there are no mechanical / moving parts. • Data is stored in floating gate transistors // data is stored in transistors that do not lose their charge/state <u>when power is no longer applied</u> // electrons are trapped between oxide layers. • Presence of trapped electrons / charge or absence indicates 0 / 1 **A.** 0 or 1 either way around **A.** state represents 0 or 1 **A.** off = 0, on = 1 (or other way around) **A.** “bit” for 0 or 1, but not “binary” **R.** positive and negative charges • NAND memory // flash memory // EEPROM memory is used. • Data is organised into pages / blocks. • A whole block (**A.** page) of data must be written // it is not possible to write individual values. • A block (**A.** page) must be erased before it can be overwritten. • Controller manages the organisation of the data // controller manages the reading and writing of data.	**4** **AO1 = 4**

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>12</td><td>1</td><td>
MP1: Values in memory locations 102 and 103 loaded into two different registers;
MP2: Comparison made between the values in the two registers;
MP3: If the values in the two registers are the same then the code will exit (after performing any other necessary instructions); A. end of program reached if not HALT instruction.
MP4: If A is greater than B then the value in the register representing B is subtracted from the value in the register representing A and result stored in register representing A;
Note: Award this mark even if further incorrect changes would also be made to values in registers.
MP5: If A is less than (or equal to B) / then the value in the register representing A is subtracted from the value in the register representing B and result stored in register representing B; Note: Award this mark even if further incorrect changes would also be made to values in registers.
MP6: A loop has been implemented which returns to the loop start after each subtraction has taken place;
MP7: Before the algorithm exits, in all circumstance, the value in the register representing A (or the register representing B) is stored into memory location 104 (regardless of whether or not this is the gcd); A. if this is done on every iteration of a loop instead of just once.
MP8: The solution correctly calculates the GCD of A and B;
DPT. Use of invalid register names eg R27, Rn
DPT. Omission of # to indicate immediate operand values
DPT. R before memory address eg R100
DPT. Use of MOV instead of LDR or STR, or vice-versa
DPT. <u>Repeated</u> use of incorrect delimiters eg ; < > . “ ‘ (occasional errors can be ignored)
Max 7 if solution not fully working
Example Solution 1
<pre>
  LDR R1, 102
  LDR R2, 103
loop:
  CMP R1, R2
  BEQ finish
  BGT agreaterthanb
  SUB R2, R2, R1
  B loop
agreaterthanb:
</pre>
</td><td>8

AO3 = 8</td></tr>
</table>

```
  SUB R1, R1, R2
  B loop
finish:
  STR R1, 104
```

Example Solution 2

```
  LDR R0, 102
  LDR R1, 103
startloop:
  CMP R0, R1
  BEQ end
  CMP R0, R1
  BGT greater
  SUB R1, R1, R0
  B startloop
greater:
  SUB R0, R0, R1
  B startloop
end:
  STR R1, 104
```

Note: Any available general purpose register (R0 to R12) may be used and any understandable method to identify a label.

DPT use of invalid register names eg R27, Rn