OXFORD AQA
INTERNATIONAL QUALIFICATIONS

INTERNATIONAL AS
COMPUTER SCIENCE

Unit 2 Concepts and principles of computer science

Specimen paper 07:00 GMT Time allowed: 1 hour 30 minutes

Materials
- You may use a calculator.

Instructions
- Use black ink or black ball-point pen.
- Fill in the boxes at the top of this page.
- Answer **all** questions.
- You must answer the questions in the spaces provided. Do not write outside the box around each page or on blank pages.
- If you need extra space for your answer(s), use the lined pages at the end of this book. Write the question number against your answer(s).
- Do all rough work in this answer book. Cross through any work you do not want to be marked.

Information
- The marks for each question are shown in brackets.
- The maximum mark for this paper is 75.

Advice
- In some questions you are required to indicate your answer by completely shading a lozenge alongside the appropriate answer as shown.
- If you want to change your answer you must cross out your original answer as shown.
- If you wish to return to an answer previously crossed out, ring the answer you now wish to select as shown.

Answer **all** questions in the spaces provided.

0 1 . 1 Shade **one** lozenge to show which Boolean operation is used by the Vernam cipher to combine the plaintext and key to produce the ciphertext.

[1 mark]

A AND ⬭

B NAND ⬭

C NOT ⬭

D OR ⬭

E XOR ⬭

0 1 . 2 Under certain circumstances, the Vernam cipher offers perfect security.

State **two** conditions that must be met for the Vernam cipher to offer perfect security.

[2 marks]

Condition 1 ______________________________

____

Condition 2 ______________________________

____

0 1 . 3 The Vernam cipher is a symmetric cipher.

Describe the difference between a symmetric and an asymmetric cipher system.

[1 mark]

____

____

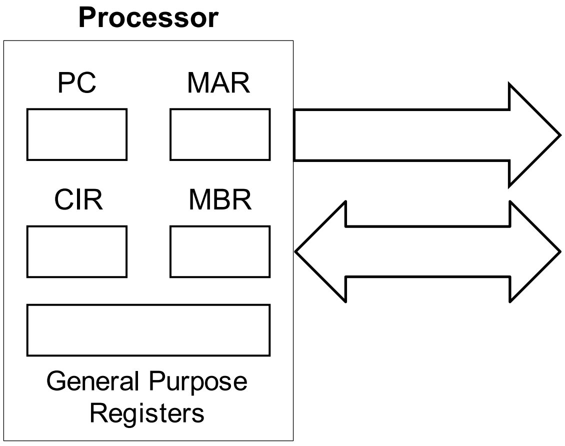
____

____

4

0 2 **Figure 1** shows some of the internal components of a processor and how the processor is connected to main memory. The internal connections within the processor are not shown.

Figure 1



Main memory

Address	Content
0	
1	
2	
3	
4	
5	
6	
7	

0 2 . 1 Describe how an instruction is fetched from main memory during the **fetch stage** of the fetch-execute cycle.

Your description should cover the use of registers, buses and main memory.

[4 marks]

The computer system shown in **Figure 1** uses the von Neumann architecture. The Harvard architecture is an alternative to this.

0 2 . 2 Explain **two** reasons why the Harvard architecture is sometimes used in preference to the von Neumann architecture.

[2 marks]

6

0 3 . 1 A sound is sampled and recorded digitally. The sound is sampled at a rate of 48 000 samples per second (Hz) for 3 minutes using a 16-bit sample resolution.

Calculate the size of the digital recording, giving your answer in mebibytes.

Give your answer rounded to 2 decimal places.

You should show your working.

[3 marks]

____

____

____

____

____

____

Answer ________________ mebibytes

0 3 . 2 The highest frequency component in a different sound is 15 000 Hz.

What is the minimum sampling rate that should be used when recording this sound to ensure that all the frequencies in the original waveform are preserved, so that when the recording is played back the original sound is recreated accurately?

[1 mark]

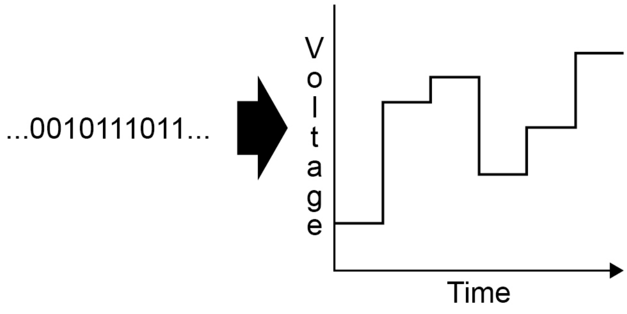
____

____

Answer ________________ Hz

0 3 . 3 **Figure 2** shows part of the process of playing back a sound that has been sampled. The binary sound data is used to generate an electrical waveform.

Figure 2



A hardware component on a sound card carries out the process shown in **Figure 3**.

State the name of this component.

[1 mark]

____

5

0 4 . 1 Explain why Unicode was introduced as an alternative to ASCII.

[2 marks]

Figure 3 shows a 7-bit ASCII character code. The character code is to be transmitted using an even parity system.

Figure 3

0	0	1	0	1	1	1

0 4 . 2 Describe how the parity bit would be generated for the character code in **Figure 3** using even parity.

[2 marks]

0 4 . 3 Write the parity bit below to complete the byte that will be sent using even parity.

[1 mark]

	0	0	1	0	1	1	1

0 4 . 4 Describe **one** limitation of the use of parity bits for managing errors.

[1 mark]

6

0 5 . 1 Define the term 'system software'.

[1 mark]

____

____

0 5 . 2 The list below contains five types of software. Four of the types are examples of system software.

Shade **one** lozenge to show which type of software is **not** system software.

[1 mark]

A Assemblers ⬭

B Bitmap image editors ⬭

C Interpreters ⬭

D Libraries ⬭

E Utility programs ⬭

0 5 . 3 Describe **two** functions of an operating system.

[2 marks]

Function 1 ____

____

____

Function 2 ____

____

____

4

0 6 . 1 A bitmap image is 1000 pixels wide by 800 pixels high.

The image takes up 400 kilobytes of storage space when represented as a bitmap, excluding metadata.

Calculate the maximum number of different colours that could appear in the image.

You should show your working.

[3 marks]

Answer ____________________ colours

0 6 . 2 The same image can also be represented using vector graphics.

The vector graphics representation of the image takes up 2 kB of storage space.

Explain why the amount of storage space taken up by the vector graphics representation of the image is significantly smaller than the space taken up by the bitmap representation.

[3 marks]

0 6 . 3 One advantage of vector graphics compared to bitmap graphics is that fewer bytes are used to represent an image.

State **two** other advantages of vector graphics compared with bitmap graphics.

[2 marks]

Advantage 1 ____

Advantage 2 ____

8

0 7 One method that can be used to improve the performance of a processor is to increase the amount of cache memory.

Describe:

- what cache memory is
- what cache memory is used for
- how increasing the amount of cache memory can improve the performance of a processor.

[4 marks]

4

Turn over for the next question

0 8 The truth table in **Table 1** represents the operation of a logic system.

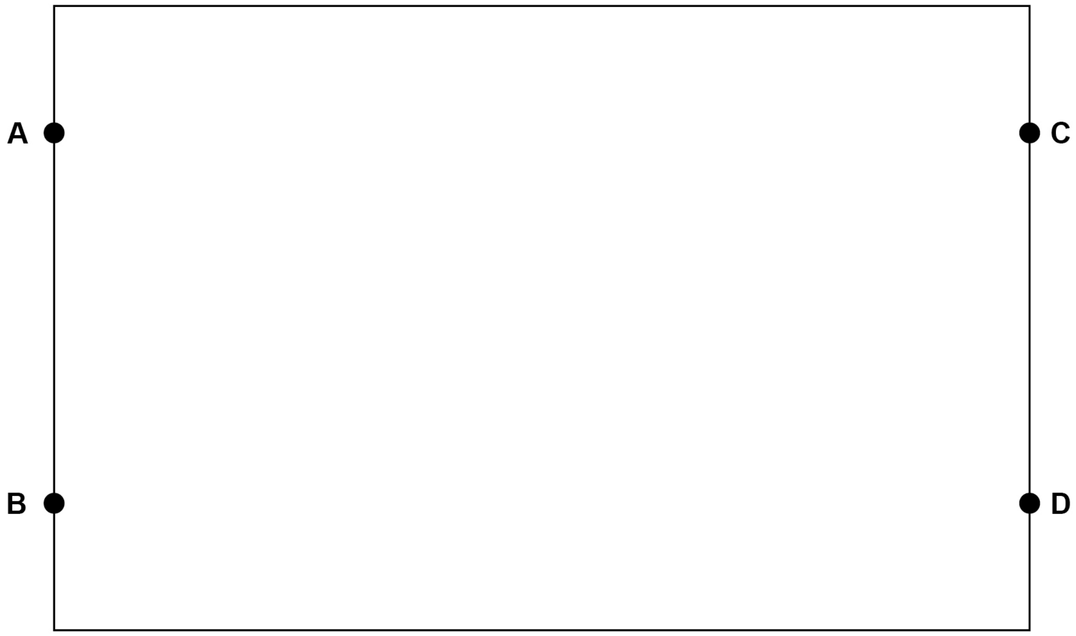
Table 1

Inputs		Outputs	
A	**B**	**C**	**D**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

0 8 . 1 In the space below, draw a logic circuit that would produce the outputs shown in **Table 1** for the given inputs.

Your circuit should only use **two** gates. If your response does not use two gates will **not** be will be able to achieve full marks.

[3 marks]



0 8 . 2 Describe the purpose of the circuit that you have drawn that produces the outputs given in **Table 1**.

[1 mark]

0 8 . 3 Using the rules of Boolean algebra, simplify the following Boolean expression.

$$A \cdot \overline{B} + B \cdot \overline{(\overline{A} + (\overline{B} \cdot C))}$$

You **must** show your working.

[4 marks]

Working ____

____

____

____

____

____

____

____

____

____

____

____

Answer ____

8

Turn over for the next question

0 9 . 1 Discuss the advantages **and** disadvantages of programming using a high-level language compared to programming using assembly language.

[4 marks]

0 9 . 2 Some compilers translate source code into an intermediate language rather than producing an executable file. Bytecode is one example of an intermediate language.

Explain how intermediate language code is used after it has been generated.

[2 marks]

0 9 . 3 State **one** reason why some compilers produce their final output in an intermediate language instead of machine code.

[1 mark]

7

1 0 A particular computer uses a **normalised** floating point representation with an 8-bit mantissa and a 4-bit exponent, both stored using **two's complement**.

Four bit patterns that are stored in this computer's memory are listed in **Figure 4** and are labelled **A**, **B**, **C** and **D**. Some of the bit patterns are valid normalised floating point numbers.

Figure 4

	Mantissa	Exponent
A	0.0000001	0000
B	0.1000000	1000
C	1.0000000	0010
D	1.1001110	1000

1 0 . 1 Shade **one** lozenge to show which bit pattern (**A–D**) in **Figure 4** represents a negative normalised value.

[1 mark]

A ⬭ **B** ⬭ **C** ⬭ **D** ⬭

1 0 . 2 Shade **one** lozenge to show which bit pattern (**A–D**) in **Figure 4** represents the smallest positive normalised value.

[1 mark]

A ⬭ **B** ⬭ **C** ⬭ **D** ⬭

1 0 . 3 The following is a floating point representation of a number:

Mantissa

1 ●	0	1	1	0	0	0	0

Exponent

0	0	1	1

Calculate the decimal equivalent of the number. You **must** show your working.

[2 marks]

____

____

____

____

Answer ______________________

1 0 . 4 Write the normalised floating point representation of the decimal value 58.5 in the boxes below. You **must** show your working.

[3 marks]

____

____

____

____

____

____

Answer

Mantissa

●							

Exponent

There can be a loss of precision when a decimal number is stored using a floating point system.

The closest possible representation of the decimal number 13.8 is shown below.

Mantissa

0	●1	1	0	1	1	1	0

Exponent

0	1	0	0

When this bit pattern is converted back to decimal its value is 13.75, not 13.8

1 0 . 5 Calculate the absolute error that has occurred.

[1 mark]

Answer ____________________

1 0 . 6 Calculate the relative error that has occurred.

Express your answer as a percentage to two decimal places.

[1 mark]

Answer ____________________

9

Turn over for the next question

1 1 . 1 Explain why desktop computers usually have secondary storage devices.

[2 marks]

1 1 . 2 A computer is fitted with a solid-state disk (SSD).

Describe the principles of operation of an SSD.

[4 marks]

6

This table is included so that you can answer Question 12.1 on page 21.

Table 2 – Standard OxfordAQA assembly language instruction set

`LDR Rd, <memory ref>`	Load the value stored in the memory location specified by `<memory ref>` into register `d`.
`LDR Rd, [Rn]`	Load the value stored in the memory location specified in register `n` into register `d`
`STR Rd, <memory ref>`	Store the value that is in register `d` into the memory location specified by `<memory ref>`.
`STR Rd, [Rn]`	Store the value that is in register `d` into the memory location specified by register `n`.
`ADD Rd, Rn, <operand2>`	Add the value specified in `<operand2>` to the value in register `n` and store the result in register `d`.
`SUB Rd, Rn, <operand2>`	Subtract the value specified by `<operand2>` from the value in register `n` and store the result in register `d`.
`MOV Rd, <operand2>`	Copy the value specified by `<operand2>` into register `d`.
`CMP Rn, <operand2>`	Compare the value stored in register `n` with the value specified by `<operand2>`.
`B <label>`	Always branch to the instruction at position `<label>` in the program.
`B<condition> <label>`	Branch to the instruction at position `<label>` if the last comparison met the criterion specified by `<condition>`. Possible values for `<condition>` and their meanings are: `EQ`: equal to `GT`: greater than `GE`: greater than or equal to `NE`: not equal to `LT`: less than `LE`: less than or equal to
`AND Rd, Rn, <operand2>`	Perform a bitwise logical AND operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.
`ORR Rd, Rn, <operand2>`	Perform a bitwise logical OR operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.
`EOR Rd, Rn, <operand2>`	Perform a bitwise logical XOR (exclusive or) operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.
`MVN Rd, <operand2>`	Perform a bitwise logical NOT operation on the value specified by `<operand2>` and store the result in register `d`.
`LSL Rd, Rn, <operand2>`	Logically shift left the value stored in register `n` by the number of bits specified by `<operand2>` and store the result in register `d`.
`LSR Rd, Rn, <operand2>`	Logically shift right the value stored in register `n` by the number of bits specified by `<operand2>` and store the result in register `d`.
`HALT`	Stops the execution of the program.

Labels: A label is placed in the code by writing an identifier followed by a colon (:). To refer to a label the identifier of the label is placed after the branch instruction.

Interpretation of `<operand2>`

`<operand2>` can be interpreted in two different ways, depending on whether the first character is a # or an R:

- `#` – use the decimal value specified after the `#`, eg `#25` means use the decimal value 25.
- `Rm` – use the value stored in register `m`, eg `R6` means use the value stored in register 6.

The available general purpose registers that the programmer can use are numbered 0 to 12.

1 2 The greatest common divisor of two positive integers A and B is the largest positive integer that divides both of the numbers without leaving a remainder.

For example, if A = 4 and B = 6 then:

- 4 has the divisors 1, 2 and 4
- 6 has the divisors 1, 2, 3 and 6

Therefore, the greatest common divisor of 4 and 6 is 2 since this is the biggest number which appears in the list of divisors of both 4 and 6.

The method shown in **Figure 5** is a famous method for determining the greatest common divisor of two positive integers, A and B:

Figure 5

```
WHILE A ≠ B
  IF A > B THEN
    A ← A - B
  ELSE
    B ← B - A
  ENDIF
ENDWHILE
```

When the algorithm terminates, the value of A is the same as the value of B, and this value is the greatest common divisor of A and B.

1 2 . 1 Write a program **using the OxfordAQA assembly language instruction set**, shown on **page 19** in **Table 2**, that uses the method described in **Figure 5** to calculate the greatest common divisor of two positive integers.

- At the start, the positive integer A will be stored in memory location 102 and the positive integer B in memory location 103. Your program should use these values to find their greatest common divisor.
- When your program terminates it should store the greatest common divisor of these two numbers in memory location 104.

[8 marks]

8

END OF QUESTIONS