

Copyright

© 2016 PG Online Limited

The contents of this unit are protected by copyright.

This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the end user, and PG Online Limited. This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it is licensed, not sold, to you by PG Online Limited for use under the terms of the licence.

The materials distributed with this unit may be freely copied and used by members of a single institution on a single site only. You are not permitted to share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are members of a separate institution. You acknowledge that the materials must remain with you, the licencing institution, and no part of the materials may be transferred to another institution. You also agree not to procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.



Teacher’s Guide

Introduction

This teacher’s guide contains a detailed lesson plan to accompany the set of PowerPoint slides and worksheets for each lesson.

The lessons are designed to form a basis for ideas for the teacher and should be adapted to suit the teaching style and preferences of the individual teacher, and the resources and nature of the individual school or Computing / ICT department.

The material supplied for this unit includes:

* 6 PowerPoint presentations, each designed to cover one or two lessons
* 6 worksheets
* 6 homework sheets
* An end-of-unit test for assessment purposes

Summary

The unit is subdivided into six topics (plus a test). Some topics may occupy more than one lesson, especially if time is taken to go over homework.

It is a theoretical unit covering all of Section 3.6 (Fundamentals of Computer Systems) in the AQA AS Level Specification 7516. The unit begins with a lesson on hardware and software and the classification of software. The role of an operating system is then covered, followed by lessons on the classification of programming languages as low-level and high-level, and programming language translators. The last two topics deal with logic gates and Boolean algebra.

Learning Outcomes for the unit

**At the end of this Unit all students should be able to:**

* define the terms hardware and software and explain the relationship between them
* Explain what is meant by system software and application software
* Describe some of the functions of operating systems and utility programs
* State with examples what is meant by high- and low-level languages
* Identify machine code and assembly code as low-level languages
* Explain why program translators are needed
* Explain the difference between source and object code
* Interpret simple assembly code programs
* construct truth tables for a variety of logic gates
* draw and interpret logic gate circuit diagrams involving multiple gates
* write a Boolean expression for a given logic gate circuit
* draw an equivalent logic gate circuit for a given Boolean expression

**Most students will be able to:**

* Explain the need for, and attributes of, different types of software
* Describe the functions of operating systems, utility programs, libraries and translators
* Describe the functions of an operating system: resource management, managing hardware to allocate processors, memories and I/O devices among competing processes
* Describe the advantages and disadvantages of machine code and assembly language programming compared with high-level programming
* Explain the difference between compilation and interpretation, and describe situations when both would be appropriate
* Write simple assembly code programs
* Use de Morgan’s laws to manipulate and simplify Boolean expressions

**Some students will be able to:**

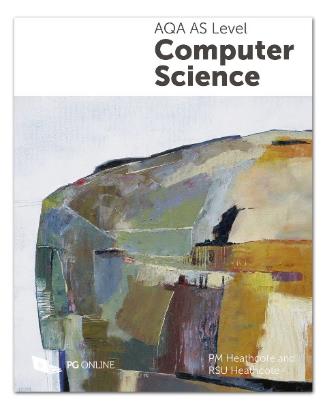
* Explain that the role of the operating system is to create a virtual machine by hiding complexities of the hardware from the user
* Explain why an intermediate language such as bytecode is produced as the final output by some compilers and how it is subsequently used
* Write assembly code programs involving arithmetic, data transfer and compare and branch instructions using a given format

Previous Learning

Students would benefit from having studied relevant material from the new KS3 National Curriculum and more specifically a Computer Science related GCSE. However, the material presented in this unit will not assume that students have studied these topics prior to this course.

Suggested Resources

No specific software is required for this unit beyond a standard office suite of applications for the presentation and printing of provided resources.



The textbook AQA AS Computer Science by PM Heathcote and RSU Heathcote, published May 2015 and reprinted with minor revisions in January 2016 (192 pages) provides comprehensive coverage of all the theoretical topics in the AQA specification (7516).

Each of the six sections in the book corresponds to one of the teaching units in this series and will be extremely useful as a course textbook and also as a revision guide. Sample questions, many taken from past exam papers, are included at the end of each chapter and can be set as additional homework.

The book is published by PG Online in a printed edition. Please refer to www.pgonline.co.uk for ordering and pricing details.

Vocabulary

Vocabulary associated with this Unit, such as:

hardware, general-purpose/special-purpose software, operating system, utility programs, defragmenter, virus checker, library program, translator, virtual machine, processor scheduling, interrupt, embedded system, machine code, assembly language, assembler, compiler, interpreter, bytecode, logic gate, truth table, Boolean algebra

Assessment

Assessment will be by means of regular homework and a test with examination style questions.

Topic plans

|  |  |  |
| --- | --- | --- |
| Topic 1 | Hardware and software |  |
| Learning Objectives:   * Define the terms hardware and software and understand the relationship between them * Explain what is meant by system software and application software * Understand the need for, and attributes of, different types of software * Understand the function of operating systems, utility programs, libraries and translators | | |
| Content | | Resources |
| **Starter**  Most students will be able to describe the difference between hardware and software. AQA uses the phrase “electrical and electro-mechanical” rather than “physical” components of a computer system.  A quick class exercise on identifying which is hardware and which is software is given with answers on the next slide.  **Main**  Go through the classification of software into system software and application software. Find out how much students know about each of these categories, and then use the slides to cover in more detail each of the system software categories.  Students should be able to briefly describe utility programs. The basic file management operations provided in Windows and other GUI operating systems such as move, copy, delete, create folder, etc. should **not** be classified as utility programs – they are an integral part of the operating system. However Windows does provide utility programs such as disk defragmentation, system restore, automatic backup, so it is something of a grey area.  Give out **Task 1** of **Worksheet 1**.  Discuss answers when students have finished.  **Libraries**  Students will probably be familiar with importing a collection of library routines into their programs, to provide routines for generating random numbers, doing graphics, etc.  Windows operating system uses hundreds of library routines which are called when needed by the OS. These have a .dll extension.  **Translators**  Students may be able to name compilers, interpreters and possibly assemblers. Answers on next slide.  **Application software**  Here we are on familiar territory. Remind students that in any question where they are asked to give a suitable type of application software, they should **not** give brand names like Word, Excel, Google Chrome etc.  Finish with a programming exercise in **Task 2** of the worksheet.The Python program is supplied and can be run by double-clicking the file name if Python is installed on the system.  **Plenary**  As a summary of the lesson, ask students to draw a chart to show all the different types of software that have been discussed in the lesson. The chart for application software was given earlier – they need to start with the division of software into system software and application software.  Give out **Homework 1**. | | PowerPoint Guide: Hardware and Software Topic 1 Hardware and software  Hardware and Software Worksheet 1  Worksheet 1 Answers  Hardware and Software Homework 1  Hardware and Software Homework 1 Answers |

|  |  |  |
| --- | --- | --- |
| Topic 2 | Role of an operating system |  |
| Learning Objectives:   * Understand the need for, and functions of an operating system * Understand that the role of an operating system is to create a virtual machine to hide the complexities of operation from the user * Understand the importance of resource management and processor scheduling | | |
| Content | | Resources |
| **Starter**  Find out what students already know about operating systems. They may name Android, Unix, or other operating systems.  **Main**  Discuss the function of the OS.  Four of the main functions of the OS are   * + Memory management   + Backing store management   + Processor scheduling   + Peripheral management of IO devices   (AQA does not require descriptions of different types of user interface, although “provision of a user interface” is one of the functions of an OS.)  Each of these functions can be described, or you could ask different groups of students to look up these four functions and tell the class what they find out.  The penultimate slide asks a question about interrupts. A printer controller will send an interrupt if it has finished printing, a keyboard controller if a user has typed some text and pressed “Enter”, a disk controller will send an interrupt if it can’t locate a particular file that has been requested (if the filename is typed wrongly, for example).  **Worksheet 2** is largely “extension tasks”, going a bit further than the specification requires, but answering these questions will help students to understand and appreciate in more detail the functions of the OS.  It is intended to provide topics for research and discussion.  **Plenary**  Discuss students’ answers to the Worksheet questions.  Give out **Homework 2**. | | PowerPoint Guide: Hardware and Software Topic 2 Role of an operating system  Hardware and Software Worksheet 2 Operating system  Worksheet 2 Answers  Hardware and Software Homework 2 Operating system  Hardware and Software Homework 2 Answers |

|  |  |  |
| --- | --- | --- |
| Topic 3 | Programming language classification |  |
| Learning Objectives:   * Be aware of the classification of programming languages into low- and high-level languages * Describe low-level languages: machine-code and assembly language * Explain the term ‘imperative high-level language’ and its relationship to low-level programming * Understand the advantages and disadvantages of machine-code and assembly language programming compared with high-level programming | | |
| Content | | Resources |
| **Starter**  The lesson starts with a brief history of the development of computer hardware and programming languages, starting with the building of Colossus at Bletchley Park in 1943.  **Main**  **Machine code**  Move on to machine code, the first programming language. In the early computers, programmers were faced with an array of, for example, 16 switches and a button to press each time they had set the switches to represent the 0s and 1s that made up the next instruction.  Show the slide with the opcode 0000 meaning  *“Load the value stored in memory location specified by the operand into the accumulator”*  Ask students to suggest some other instructions that would be useful. “STORE” is an obvious one, as well as ADD, SUBTRACT, MULTIPLY DIVIDE. The latter two are not easily implemented in machine code so are generally performed with shifts and addition, and not included in the basic instruction set shown here. Branches are necessary to perform loops and IF statements.  Hand out **Worksheet 1, Task 1** which will give the students a good idea of what programming in machine code is all about  **Assembly language**  Go through the examples of Assembly code. Ask what the advantages are, over machine code. (Answers on next slide)  **High-level languages**  FORTRAN (FORmula TRANslation) was the first high-level language, followed by ALGOL (ALGOrithmic Language), COBOL (COmmon Business Oriented Language), BASIC (Beginners All-purpose Symbolic Instruction Code) and many others.  Elicit the meaning of “high-level” as in “imperative high level languages”  Hand out **Task 2** of the worksheet and ask students to complete the questions.  Discuss answers – advantages and disadvantages of high-level languages are given on subsequent slides.  **Plenary**  The final slide describes briefly the extraordinary achievement of landing men on the moon using a basic computer with 64K of memory. How much memory do the school computers have?  Give out **Homework 3**. | | PowerPoint Guide: Hardware and Software Topic 3 Programming language classification  Hardware and Software Worksheet 3 Programming language classification  Worksheet 3 Answers  Hardware and Software Homework 3 Programming language classification  Hardware and Software Homework 3 Answers |

|  |  |  |
| --- | --- | --- |
| Topic 4 | Programming language translators |  |
| Preparation:  This is a short lesson, so you may want to combine it with some practical work or going over homework questions.  Learning Objectives:   * Understand the role of an assembler, compiler and interpreter * Explain the difference between compilation and interpretation, and describe situations when each would be appropriate * Explain why an intermediate language such as bytecode is produced as the final output by some compilers and how it is subsequently used * Understand the difference between source and object (executable) code | | |
| Content | | Resources |
| **Starter:**  Start by finding out how much students already know about program translators.   * What happens to a program written by a programmer in Assembly language before it can be executed? * What is this process called? * What is the code written by the programmer known as? (source code) * What is the output from the assembler known as? (object code) * Can the object code be saved and run whenever needed? (Yes)   Use the first two slides to recap these points  **Main:**  Move on to discuss the uses of assembly code. Make sure that the students understand that “source code” is the general term for the code that the programmer writes, whatever language is being used. “Object code” is produced by the translator, whether that is an assembler or a compiler.  **Compilers and interpreters**  Compilers are one form of programming language translator used to translate a high-level language. The compiler will find syntax errors or missing subroutines, for example, but not logic errors. Ask students for examples of each of these types of error.  Examples of errors that a compiler will find include:  result = num2\*3 (assuming num2 has not yet been given a value)  or choice = prnt(“Hello”) (misspelling of print)  Examples of logic errors include:  Performing a while loop too many or too few times  Using a plus sign in a calculation when a minus sign was intended.  Contrast compilers and interpreters. Discuss the advantages of each. Have students used both these translators in their practical work?  An **interpreter** is useful for program development but as it does not produce object code, it cannot be used for translating software that is to be distributed or sold. The user must have the interpreter present to run the code.  **Portability** is an important feature of high-level languages. With only minor modifications, source code written for one type of computer can be compiled and run on a different type of computer. This is in contrast to assembly code, which is specific to a particular machine architecture.  **Bytecode**  Bytecode is an intermediate form between source code and object code produced by some programming languages such as Python and Java.  It has the advantage of being portable between different platforms or processor architectures.  According to Wikipedia*, “The name bytecode stems from instruction sets which have one-byte* [*opcodes*](https://en.wikipedia.org/wiki/Opcode) *followed by optional parameters. Intermediate representations such as bytecode may be output by programming language implementations to ease interpretation, or it may be used to reduce hardware and operating system dependence by allowing the same code to run on different platforms. Bytecode may often be either directly executed on a virtual machine (i.e. interpreter), or it may be further compiled into machine code for better performance.”*  Give out **Worksheet 4** and ask students to do the questions.  **Plenary**  Discuss their answers, and recap the main points covered in the final slide.  Give out **Homework 4**. | | PowerPoint Guide: Hardware and Software Topic 4 Programming language translators  Hardware and Software Worksheet 4 Programming language translators  Worksheet 4 Answers  Hardware and Software Homework 4 Programming language translators  Hardware and Software Homework 4 Answers |

|  |  |  |
| --- | --- | --- |
| Topic 5 | Logic gates |  |
| Learning Objectives:   * Construct a truth table for a variety of logic gates * Be familiar with drawing and interpreting logic gate circuit diagrams involving multiple gates * Complete a truth table for a given logic gate circuit * Write a Boolean expression for a given logic gate circuit * Draw an equivalent logic gate circuit for a given Boolean expression | | |
| Content | | Resources |
| **Starter:**  Hold a brief discussion to find out what students know about how circuits are built – if you have a printed circuit board to pass around this could be useful. Basically circuits are built of switches which can either be ON or OFF. These switches are called **logic gates**. Different types of logic gate can be combined to create complex circuits. The concept of Boolean logic will be familiar to students through programming IF .. THEN statements.  e.g. IF (x = 1) AND (y = 2) THEN (do something)  IF (x = 0) OR (y = 0) THEN (do something)  What will be the outcome, TRUE or FALSE, for different values of x and y in each case?  **Main:**  There are six different logic gates that are studied in this unit. For each of them, we can write a truth table to show what happens for different combinations of TRUE or FALSE, 1 or 0, ON or OFF.  **NOT gate**  Show the slide with the first gate, the NOT gate. This has only one input and outputs 1 if the input is zero, and vice versa.  Make sure to explain carefully the three different ways of representing a logic gate; as a graphical symbol, a Truth table, and as Boolean algebra  Boolean algebra has its own set of symbols which the students need to know. The overbar represents NOT.  Commonly, inputs are given as A, B, C etc. and outputs as P, Q, R, S etc.  **AND Gate**  Move on to the AND gate and its representation. The symbol for AND in Boolean algebra is • or “.” (a full-stop), or nothing at all e.g. ‘AB’.  Explain the truth table in terms of everyday statements. For example:  A learner driver may drive a car IF (A) they have a learner’s licence AND they are accompanied by a fully licenced driver. What is the outcome (P) for each combination of A and B?  You can use the example of two taps connected in series – water can only flow if both taps are ON. Or, an electric current can only flow IF both switches are ON.  **OR Gate**  Next is the OR gate. Note the graphical symbol and the Boolean notation which rather confusingly uses + for “OR”.  Drawn as a circuit, this would be two switches in parallel:  If one or both of the switches are closed, (TRUE) current will flow.  Now you can show how two logic gates can be combined. This can be done as a class exercise. Can they write the Boolean notation? (Answer on next slide.)  **Increasing the number of inputs**  There may be several inputs, each going into a particular gate. The combination of gates produces a particular output. For three gates there are 23 possible combinations of input.  Show the next slide and get the class to complete the truth table as a class exercise. Start each column with a 0 for consistency. The binary values in rows 1-8 should count up to represent 000-111 (or 010-710).  Give out **Worksheet 5** and let students work through the questions in **Task 1**.  Go over the answer when they have finished.  The final three gates are XOR (Exclusive OR), NAND (NOT AND) and NOR (NOT OR). Go through each of these in turn.  **The universal gates**  Any gate can be replaced by a combination of NAND gates or NOR gates.  Show the next slide which demonstrates how a NOT gate can be replaced by a NAND gate with a single input A going into both the input pins. (Question 3 of **Task 2** on the worksheet shows how to replace an AND gate with two NAND gates.)  Manufacturing logic circuits is sometimes cheaper if only one type of gate is used, even if it means more gates. On the other hand, using as few gates as possible can speed up processing.  Ask students to work on the questions in **Task 3**.  **Plenary**  Go over the questions and give out homework. | | PowerPoint presentation  Hardware and Software Topic 5.ppt  Hardware and Software Worksheet 5.docx  Hardware and Software Worksheet 5 Answers.docx  Hardware and Software Homework 5.docx  Hardware and Software Homework 5 Answers.docx |

|  |  |  |
| --- | --- | --- |
| Topic 6 | Boolean algebra |  |
| Learning Objectives:   * Be familiar with the use of Boolean identities and De Morgan’s laws to manipulate and simplify Boolean expressions * Write a Boolean expression for a given logic gate circuit, and vice versa | | |
| Content | | Resources |
| **Starter**  Introduce Augustus de Morgan, who formulated the laws which will be studied in this topic. There is an interesting biography at <https://en.wikipedia.org/wiki/Augustus_De_Morgan> and many other websites.  **Main**  Move on to de Morgan’s first law,  You can ask the students to make up sentences to show that this is correct – instead of saying, for example, “He’s not in the house and he’s not in the garden” we could say “He’s not in the house or the garden”. “I don’t like turnips and I don’t like leeks” is the same as “I don’t like turnips or leeks”…  Students may be familiar with VENN diagrams, which show the truth of de Morgan’s laws in a graphical way.  Truth tables for each side of the equation can be used to verify that the law is true. The students can complete the Truth tables as a class exercise. (Completed tables on next slide)  Go over de Morgan’s second law in a similar way. These laws are very useful for simplifying Boolean expressions.  **Simplifying expressions**  There are a number of rules which can be used in simplifying expressions. Most of these are quite intuitive if students think carefully about what they actually mean – the rules need to be understood rather than memorised. It may be helpful to ask them to make up sentences to demonstrate each rule. For example, Any TRUE statement AND a false statement must be false, i.e. 0, e.g. “The moon orbits the earth AND the moon is made of cheese” illustrates Rule 1. “A cat is an animal AND a cat is not an animal” illustrates that rule 4 must always be false, i.e. 0.  The right-hand sides of each rule are displayed one by one on a series of slides – get the students to figure each one out first, and make up a statement illustrating it, before displaying the answer.  **Commutative, associative and distributive rules**  These work much like mathematics. The last one (numbered 15) is similar to multiplying two bracketed expressions.  (X + Y).(W + Z) = X.W + X.Z + Y.W + Y.Z  **Absorption rules**  These rules are two of the most useful for simplifying expressions. *If stuck, see if the Absorption rule can be applied.*  Give out **Worksheet 6**, **Task 1**, which gives practice in using these rules to simplify expressions.  Go over the answers when students have finished Task 1.  **Drawing logic diagrams**  Remind students of the NAND gate introduced in the last lesson, and how NOT (A AND B) can be represented in two ways. The same is true of the NOR gate.  Ask them to draw logic diagrams for and  The answers are on the next slide. Note that the equation  is represented using a NAND gate, and the equation representing is represented using a NOR gate, using de Morgan’s Laws.  On the next slide, students can practise finding the Boolean expression that represents a given logic circuit. The output from each gate should be written in and then these can be combined to give the final output.  This can often be simplified so that the same output can be achieved with fewer gates. This is shown on the next slide.  Before students attempt the questions in Task 2 of the worksheet, point out that the same input can be used going into two different gates. The output from the gates shown on the Worksheet slide is A + AB.  Ask students to work on questions in Task 2, and go over the answers when they have finished.  **Plenary**  Some students may find this rather mathematical topic quite difficult. Emphasise that it is all a matter of practice!  Give out **Homework 6**. | | PowerPoint presentation  Hardware and Software Topic 6.ppt  Hardware and Software Worksheet 6.docx  Hardware and Software Worksheet 6 Answers.docx  Hardware and Software Homework 6.docx  Hardware and Software Homework 6 Answers.docx |

|  |  |
| --- | --- |
| Unit assessment | |
| Learning Outcomes:  Students will   * apply their knowledge in answers to a range of questions * be able to highlight areas of strength and any gaps in their understanding of computers | |
| Content | Resources |
| Students should complete the **Assessment Test**.  These tests have been designed to be printed and answered by hand. | Hardware and software Unit assessment  Hardware and software Unit assessment Answers |

Acknowledgements

The authors and publisher would like to thank all contributors for their kind permission to reproduce their photographs or images, screenshots of their websites or other copyright material in the PowerPoint Guides.

Images unless otherwise stated © Shutterstock

Every effort has been made to trace the copyright holders and we apologise in advance for any unintentional omissions. We would be pleased to insert the appropriate acknowledgement in any subsequent edition of this publication.

This material contains links to relevant websites featured in the teacher’s guide. Every effort has been made to ensure that at the time of distribution, the links remain unbroken, the material remains up-to-date and that links are not inadvertently linked to sites that could be considered offensive. PG Online cannot be held responsible for the content of any website mentioned in this material. It is sometimes possible to find relocated sites by typing the original URL into a browser. Any errors should be reported directly to support@pgonline.co.uk and changes will be made in any subsequent editions of the material.

Artwork



A year on my moor © 2015 Heather Duncan

Oil on board, 60x60cm

www.heatherduncan.com