Accredited

A LEVEL

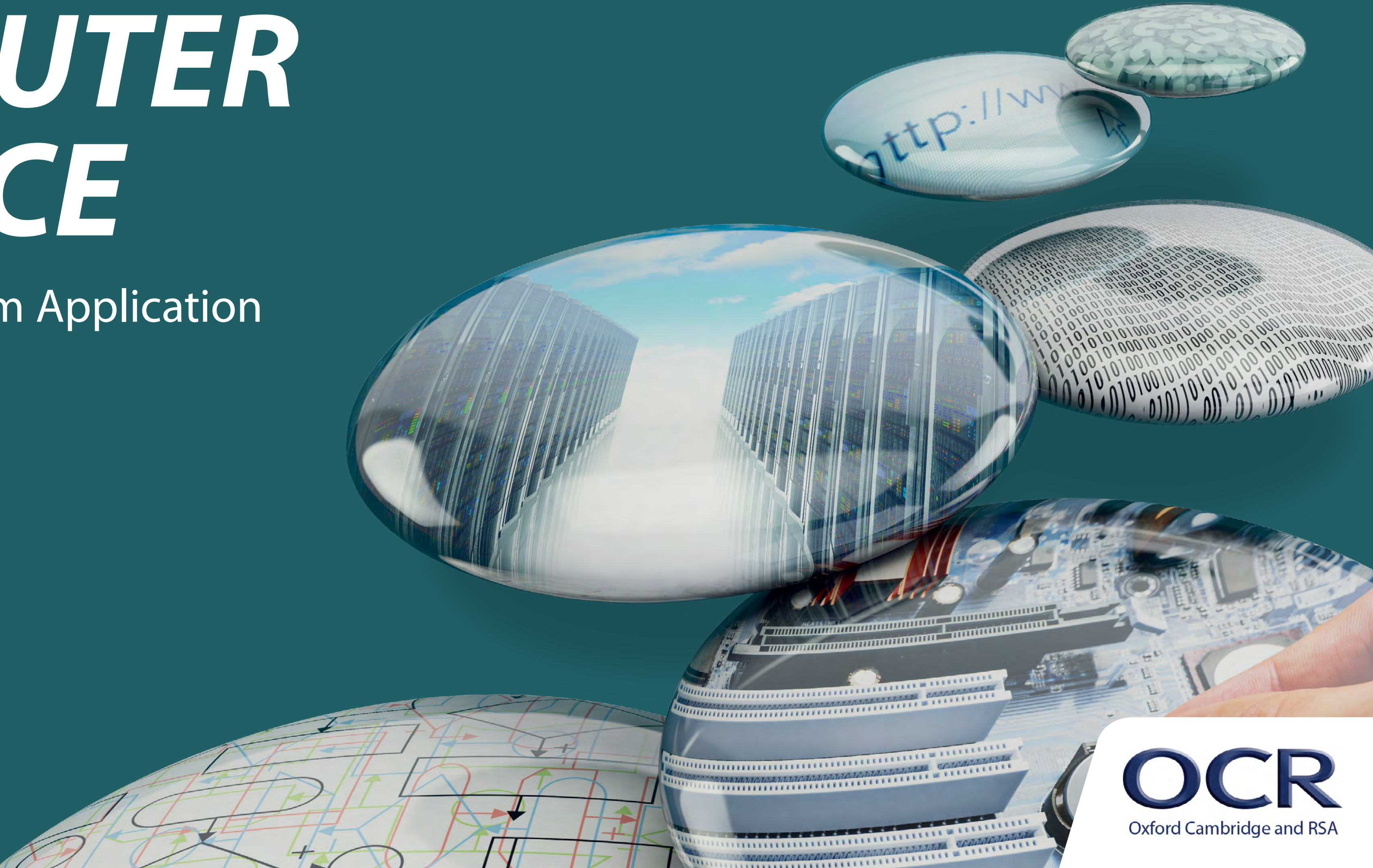
Delivery Guide

H446

COMPUTER SCIENCE

Theme: 1.2.2 System Application Generation

June 2015



Oxford Cambridge and RSA

We will inform centres about any changes to the specification. We will also publish changes on our website. The latest version of our specification will always be the one on our website (www.ocr.org.uk) and this may differ from printed versions.

Oxford Cambridge and RSA Examinations is a Company Limited by Guarantee.
Registered in England. Registered company number 3484466.

Registered office: 1 Hills Road
Cambridge
CB1 2EU

OCR is an exempt charity.

CONTENTS

Introduction

Delivery guides are designed to represent a body of knowledge about teaching a particular topic and contain:

- Content: a clear outline of the content covered by the delivery guide;
- Thinking Conceptually: expert guidance on the key concepts involved, common difficulties students may have, approaches to teaching that can help students understand these concepts and how this topic links conceptually to other areas of the subject;
- Thinking Contextually: a range of suggested teaching activities using a variety of themes so that different activities can be selected that best suit particular classes, learning styles or teaching approaches.

If you have any feedback on this Delivery Guide or suggestions for other resources you would like OCR to develop, please email resources.feedback@ocr.org.uk.

KEY

🗋	Click to view associated resources within this document.
▶ Click here	Click to view external resources
AS Level only	AS Level content only

Curriculum Content

a) The nature of applications, justifying suitable applications for a specific purpose.

b) Utilities.

c) Open source vs Closed source.

d) Translators: Interpreters, compilers and assemblers.

e) Stages of compilation (Lexical Analysis, Syntax Analysis, Code Generation and Optimisation).

f) Linkers and loaders and use of libraries.

Thinking Conceptually

Approaches to teaching the content

Much of what students need to understand about this area of the specification can make good use of discussion, although the later sections about compilers and interpreters won't necessarily be things that students have come across before.

The nature of applications, justifying suitable applications for a specific purpose	Resources
Students should be familiar with applications but may have not considered that there are alternatives and if so, what the differences are. This could provide the basis for an interesting discussion about apps on smartphone vs traditional applications on PCs, Free vs Paid, advertising and alternative pieces of software such as Microsoft Word vs Libre Office, Google Docs, Pages (Mac) and Evernote, just to name a few. What categories do these pieces of software fall under? Are some categories newer than others? Which pieces of software need certain hardware to function (e.g. speed, memory, cameras etc)? The website http://download.cnet.com/windows/ can be used by students to get inspiration. Who would use different categories of software? http://en.wikipedia.org/wiki/Application_software http://www.teach-ict.com/as_as_computing/ocr/H047/F451/311/c_application_sw/miniweb/index.htm	▶ Click here ▶ Click here ▶ Click here
Utilities	**Resources**
Utilities differ from general applications as they are usually focused on system maintenance such as firewalls, antivirus, backup, data compression or defragmentation, for example. The same sort of teaching methods as above can be used for utilities. A video that covers most utility software can be found here: https://www.youtube.com/watch?v=_PAsXKddNF4 http://en.wikipedia.org/wiki/Utility_software	▶ Click here ▶ Click here

Thinking Conceptually

Open source vs Closed source

Open source software is that for which the source code is free and the user can edit the source code and can potentially contribute to the community to improve the piece of software. Linux is an example of this, since it is free and open source. Closed source is where the company that owns the source code keeps it to itself. The software might be free or need to be paid for. Microsoft works in this way.

You could watch some of the videos below before having a class discussion on which type of software students think is better in the long run – closed or open source? You could get students into groups and let them prepare their argument for or against.

Some engaging videos that cover the basics using practical examples can be found at: https://www.youtube.com/watch?v=Tyd0FO0tko8 and https://www.youtube.com/watch?v=a8fHgx9mE5U

http://en.wikipedia.org/wiki/Comparison_of_open_source_and_closed_source

▶ Click here

▶ Click here

▶ Click here

Translators: Interpreters, compilers and assemblers

Resources

A translator is a program that translates a computer program written in one language into another which functions exactly the same. This is commonly used since computers don’t understand programs written in high level languages like Java and Python and so something is needed to translate these into machine code that the computer understands and can run.

http://en.wikipedia.org/wiki/Translator_(computing)

Assemblers are programs that translate an assembly language program into machine code so that a computer can understand it. Assemblers do not ‘make’ any executable files.

Compilers do the same sort of basic thing as assemblers, except they are more advanced as they can perform operations such as code optimisation and they compile straight from the source language (e.g. Java) into the target language (machine code).

Interpreters are different from compilers because instead of compiling all of a program into machine code at once it does so one statement at a time.

http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_2/translators_compilers/miniweb/index.htm

Video – What is Compiler and Interpreter (in easiest way) (Scott Barron): http://community.computingatschool.org.uk/resources/1711 also at https://www.youtube.com/watch?v=kmQUB-5cEgM

▶ Click here

▶ Click here

▶ Click here

▶ Click here

Thinking Conceptually

Stages of compilation (Lexical Analysis, Syntax Analysis, Code Generation and Optimisation)	Resources
Lexical analysis is responsible for gathering all the source code and 'tokenising' it. This means that it will be able to understand which are variables, which are keywords (such as if, else etc) and also remove comments. Syntax analysis ensures that the code makes logical sense. For instance, if a bracket has been missed out, this is usually the stage where it would warn the user after processing its checks. Code generation is where the computer converts the high level code that has been written by the user into code that is understood by the computer (machine code). This process may also contain some optimisation. http://en.wikipedia.org/wiki/Compiler#Structure_of_a_compiler http://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_2/lexical_syntax_analysis/miniweb/index.htm An activity here may be to write the different stages on the board jumbled up and students have to put them into the right order and explain what each one does.	▶ Click here ▶ Click here
Linkers and loaders and use of libraries	**Resources**
Linkers take object files and required libraries and combine them into a single executable file: http://en.wikipedia.org/wiki/Linker_(computing) Loaders place programs and libraries into memory and so prepare a program for execution. http://en.wikipedia.org/wiki/Loader_(computing) A good overview of Assembler, Compiler, Interpreter, Linker, Loader can be found here: http://vijaybhargav05bop.blogspot.co.uk/2013/03/assembler-compiler-interpreter-linker.html	▶ Click here ▶ Click here ▶ Click here

Thinking Conceptually

Common misconceptions or difficulties students may have
There are quite a few different terms that students may not have encountered before in this topic. A student glossary is an excellent idea; students can add terms that are unfamiliar to them and define them in their own words.

Keywords: Open source, Closed source, Translator, Interpreter, Compiler, Assembler, Lexical analysis, Tokenisation, Syntax analysis, Code generation, Optimisation, Linker, Loader, Library

Conceptual links to other areas of the specification – useful ways to approach this topic to set students up for topics later in the course
This section of the specification is probably best taught before moving onto 1.2.4 Types of Programming Language since it covers what happens during compilation or interpretation of different languages. This would make it easier for students to understand some of the differences and similarities between languages, and also how these languages are understood by the computer.

Thinking Contextually

Activities	Resources
Activity 1 – Different types of software *Part 1* Students should use the set of cards below to try and find different pieces of software that fit into those categories, using the table to record their findings. This should expand students' ideas about what software is out there and how they are categorised. *Part 2* Taboo is a fun game where one person has to sit at the front of their group and they have a set of things that they must get the rest of their group to guess without using any of the words in the answer. For example, for Microsoft Word you might say "Bill Gates letter writing software". You could divide students up into teams, and if anyone is a bit reluctant they could always be the judge/scorekeeper.	Learner Resource 1
Activity 2 *Part 1* Using the terms on the cards below, make a powerpoint presentation with a word per slide about what each of the terms means. You could get students to work in groups, and then you have the choice of getting them to present it. Students could rate presentations and choose parts of the best ones to feed into Part 2. *Part 2* Make a mindmap that uses all the words from the cards below, trying to reduce the explanation of the terms used to the bare minimum. This could be done as a class or in groups so that each student has to come up with a picture and short description for each word, which should make the process much faster.	Learner Resource 2
Activity 3 – Compiler vs Interpreter (Digital schoolhouse) http://www.resources.digitalschoolhouse.org.uk/key-stage-4-ages-15-16/215-compiler-vs-interpreter An interesting activity based on the idea of the United Nations to help students understand the differences between compilers and interpreters. You will need to print out the 12 role cards given on the document from the website and students will have to arrange themselves into countries A, B and C and read their cards out in order from 1–12. Questions are provided to help students draw out reasonings as to how interpreters differ from compilers.	▶ Click here

Learner Resource 1 Activity 1 cards

See page 10

Security software	Browsers	Business software	Communications	Desktop enhancements
Developer tools	Digital photo software	Drivers	Educational software	Entertainment software
Games	Graphic design software	Audio software	Video software	

Learner Resource 1 Activity 1

Software name	Type of software	Application or utility?	How much does it cost?	Open or closed source?	What platform is it available for?	Is it special in any way compared to other software? Why?	Who might use this piece of software? What would they use it for?

Learner Resource 2 Activity 2

See page 10

Open source	Closed source	Translator	Interpreter	Compiler
Assembler	Lexical analysis	Syntax analysis	Code generation	Optimisation
Linker	Loader	Library		

👍 👎

We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

If you do not currently offer this OCR qualification but would like to do so, please complete the Expression of Interest Form which can be found here:
http://www.ocr.org.uk/qualifications/expression-of-interest/

OCR Resources: *the small print*
OCR's resources are provided to support the teaching of OCR specifications, but in no way constitute an endorsed teaching method that is required by the Board and the decision to use them lies with the individual teacher. Whilst every effort is made to ensure the accuracy of the content, OCR cannot be held responsible for any errors or omissions within these resources. We update our resources on a regular basis, so please check the OCR website to ensure you have the most up to date version.

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications: resources.feedback@ocr.org.uk

OCR customer contact centre

General qualifications
Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

LLOYD'S REGISTER • LRQA
ISO 9001
UKAS MANAGEMENT SYSTEMS
001

A DIVISION OF
CAMBRIDGE ASSESSMENT

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.