GCE AS

eduqas
Part of WJEC

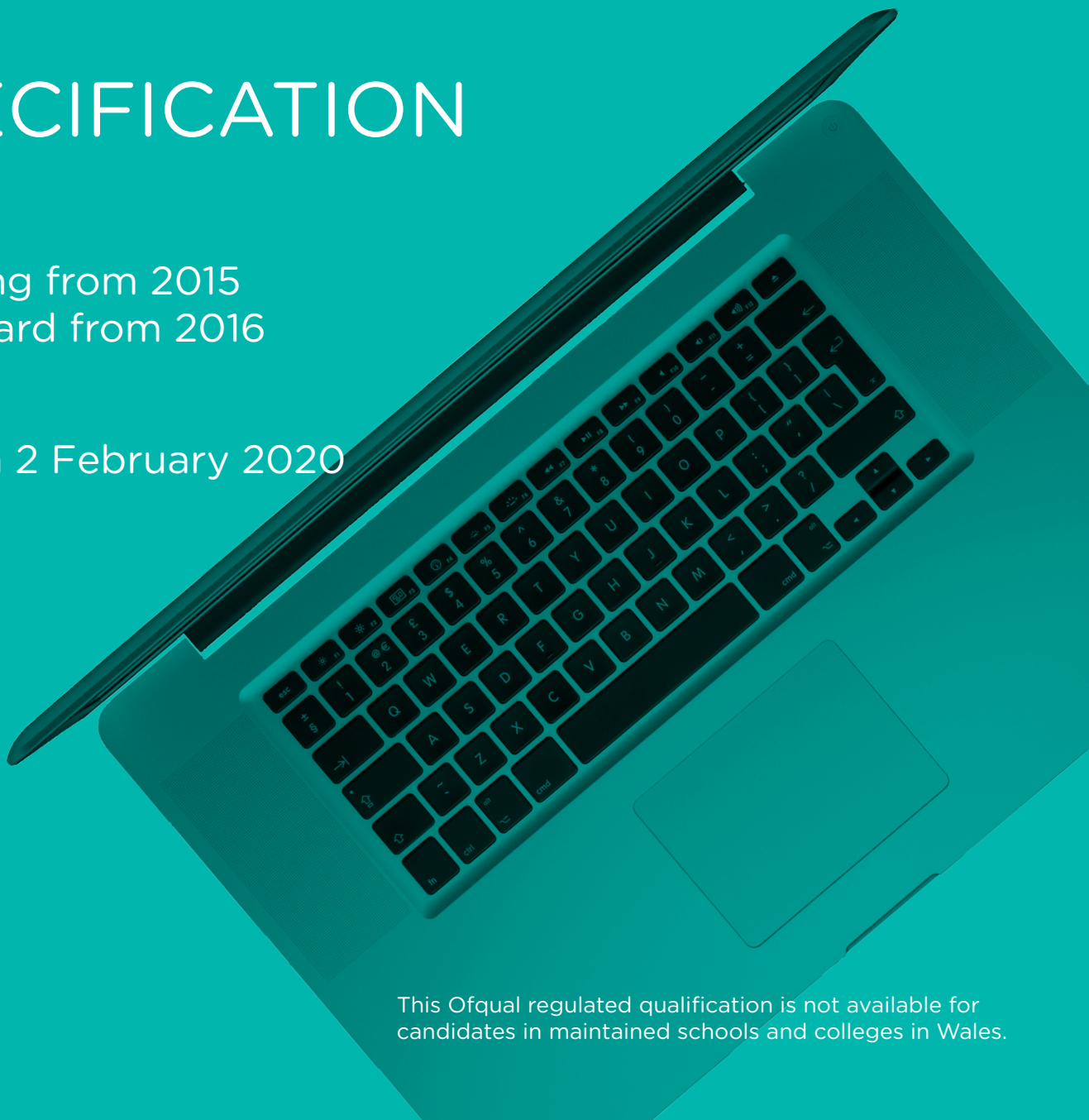
WJEC Eduqas GCE AS in COMPUTER SCIENCE

ACCREDITED BY OFQUAL

SPECIFICATION

Teaching from 2015
For award from 2016

Version 2 February 2020



wjec
cbac

This Ofqual regulated qualification is not available for candidates in maintained schools and colleges in Wales.

SUMMARY OF AMENDMENTS

Version	Description	Page number
2	'Making entries' section has been amended to clarify resit rules.	16

WJEC Eduqas GCE AS in COMPUTER SCIENCE

For teaching from 2015
For award from 2016

AS COMPUTER SCIENCE
SUMMARY OF ASSESSMENT

Component 1: Fundamentals of Computer Science
Written examination: 2 hours
70% of qualification

This component investigates computer architecture, communication, data representation, data structures, software applications, programs, algorithms, logic, programming methodologies and the impact of computer science on society.

Component 2: Practical Programming to Solve Problems
On-screen examination: 2 hours 15 minutes.
30% of qualification

This component consists of a series of set tasks completed on-screen by candidates. These tasks will assess the practical application of knowledge and understanding and will require the use of Visual Basic.NET, Python or Java as a programming language.

This linear qualification will be available in the summer series each year. It will be awarded for the first time in summer 2016.

Qualification Accreditation Number: 601/5302/7

AS COMPUTER SCIENCE

1 INTRODUCTION

1.1 Aims and objectives

The WJEC Eduqas AS in Computer Science encourages learners to develop:

- an understanding of, and the ability to apply, the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms and data representation
- the ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so
- the capacity for thinking creatively, innovatively, analytically, logically and critically
- the capacity to see relationships between different aspects of computer science
- mathematical skills – see Appendix A
- the ability to articulate the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology.

Computers are widely used in all aspects of business, industry, government, education, leisure and the home. In this increasingly technological age, a study of computer science, and particularly how computers are used in the solution of a variety of problems, is not only valuable to the learners but also essential to the future well-being of the country.

Computer science integrates well with subjects across the curriculum. It demands both logical discipline and imaginative creativity in the selection and design of algorithms and the writing, testing and debugging of programs; it relies on an understanding of the rules of language at a fundamental level; it encourages an awareness of the management and organisation of computer systems; it extends the learners' horizons beyond the school or college environment in the appreciation of the effects of computer science on society and individuals. For these reasons, computer science is as relevant to a learner studying arts subjects as it is to one studying science subjects.

The WJEC Eduqas AS in Computer Science has been designed to give an understanding of the fundamental concepts of computer science and a broad scope of study opportunities. This specification has been designed to free centres to concentrate on innovative delivery of the course by having a streamlined, uncomplicated, future-proof structure, with realistic technological requirements.

1.2 Prior learning and progression

There are no prior learning requirements. Any requirements set for entry to a course following this specification are at the discretion of centres. It is reasonable to assume that many learners have achieved qualifications equivalent to Level 2 at KS4. Skills in Numeracy/Mathematics, Literacy/English and Information Communication Technology will provide a good basis for progression to this qualification.

Some learners will have already gained knowledge, understanding and skills through their study of Computer Science at GCSE.

Mathematical requirements are specified in the subject criteria and repeated in Appendix A of this specification.

This specification provides a suitable foundation for the study of Computer Science at A level. In addition, the specification provides a coherent, satisfying and worthwhile course of study for learners who do not progress to further study in this subject.

This specification is not age specific and, as such, provides opportunities for candidates to extend their life-long learning.

1.3 Equality and fair assessment

This specification may be followed by any learner, irrespective of gender, ethnic, religious or cultural background. It has been designed to avoid, where possible, features that could, without justification, make it more difficult for a learner to achieve because they have a particular protected characteristic.

The protected characteristics under the Equality Act 2010 are age, disability, gender reassignment, pregnancy and maternity, race, religion or belief, sex and sexual orientation.

The specification has been discussed with groups who represent the interests of a diverse range of learners, and the specification will be kept under review.

Reasonable adjustments are made for certain learners in order to enable them to access the assessments (e.g. candidates are allowed access to a Sign Language Interpreter, using British Sign Language). Information on reasonable adjustments is found in the following document from the Joint Council for Qualifications (JCQ): *Access Arrangements and Reasonable Adjustments: General and Vocational Qualifications.*

This document is available on the JCQ website (www.jcq.org.uk). As a consequence of provision for reasonable adjustments, very few learners will have a complete barrier to any part of the assessment.

2 SUBJECT CONTENT

This specification promotes the integrated study of computer science. It will enable learners to develop a broad range of skills in the areas of programming, system development, computer architecture, data, communication and applications.

The knowledge, understanding and skills are set out in the two columns in the pages which follow. The topic to be studied is in the first column, with the amplification in the second column. There is no hierarchy implied by the order in which content and amplification are presented, nor should the length of the various sections be taken to imply any view of their relative importance.

The subject content for AS Computer Science will be assessed across 2 components.

Component 1 – Fundamentals of Computer Science
Written examination – 2 hours
70% of qualification

Component 2 – Practical Programming to Solve Problems
On-screen examination – 2 hours 15 minutes
30% of qualification

2.1 Component 1 and Component 2

1. Hardware and communication

	Identify and describe the hardware and communication elements of contemporary computer systems and how they are connected.
Architecture	Identify and describe the main components of contemporary computer architecture, including Von Neumann architectures.
	Describe different types of memory and caching.
	Describe and explain parallel processing.
Fetch-execute cycle	Describe the fetch-execute cycle showing how data can be read from RAM into registers.
Input / output	Describe the use of contemporary methods and their associated devices for input and output.
	Explain the use of these methods and devices in contemporary computer systems and their suitability in different situations.
Secondary storage	Compare the functional characteristics of contemporary secondary storage devices.

Data storage on disc	Explain fragmentation and its consequences and describe the need for defragmentation.
Networking	Describe networks and how they communicate.
	Explain the importance of networking standards.
	Describe the importance and the use of a range of contemporary protocols including HTTP, FTP, SMTP, TCP/IP, IMAP, DHCP, UDP and wireless communication protocols.
	Explain the role of handshaking.
Internet	Describe the internet in terms of a world-wide communications infrastructure.

2. Logical operations

	Draw truth tables for Boolean expressions consisting of AND, OR, NOT and XOR logical operations.
	Apply logical operations to combinations of conditions in programming, including clearing registers and masking.
	Simplify Boolean expressions using Boolean identities and rules.

3. Data transmission

	Describe serial and parallel transmission, their advantages and disadvantages.
	Describe simplex, half duplex and full duplex transmission methods.
	Explain the need for multiplexing and switching.
Communication networks	Describe, using appropriate network protocols, such as TCP/IP the typical contents of a packet.
	Explain network collision, network collision detection and how these collisions are dealt with.
	Describe methods of routing traffic on a network.

4. Data representation and data types

Representation of data as bit patterns	Explain the terms bit, byte and word.
	Describe and use the binary number system and the hexadecimal notation as shorthand for binary number patterns.

Storage of characters	Describe how characters and numbers are stored in binary form.
	Describe standardised character sets.
Data types	Describe the following different primitive data types, Boolean, character, string, integer and real.
	Describe the storage requirements for each data type.
Representation of numbers as bit patterns	Apply binary arithmetic techniques.
	Explain the representation of positive and negative integers in a fixed-length store using both two's complement, and sign and magnitude representation.
	Describe the nature and uses of floating point form.
	State the advantages and disadvantages of representing numbers in integer and floating point forms.
	Convert between real number and floating point form.
	Describe truncation and rounding and their effect upon accuracy.

5. Data structures

	Describe, interpret and manipulate data structures including arrays (up to two dimensions) and records.
	Describe the manipulation of records and arrays.
	Select, identify and justify appropriate data structures for given situations.

6. Organisation of data

	Explain the purpose of files in data processing.
File design	Define a file in terms of records and fields.
	Explain fixed and variable length fields and records and give examples of the appropriate use of each type.
	Design files and records appropriate for a particular application.

File organisation	Distinguish between master and transaction files.
	Describe sequential, indexed sequential and direct (random) file access.
	Distinguish between the use of serial and sequential file access methods in computer applications.
	Describe and design algorithms and programs for sequential file access and update.
	Explain the need for file security, including file backup, generations of files and transaction logs.
	Describe the need for archiving files.
Data validation and verification	Explain and apply appropriate techniques for data validation and verification.
	Design algorithms and programming routines that validate and verify data.

7. Database systems

	Describe and discuss the benefits and drawbacks of relational database systems and other contemporary database systems.
	Describe the use of primary and foreign keys, indexes and links.
	Explain and apply entity relationship modelling and use it to analyse simple problems.
	Describe the advantages of different users having different views of the data in a database.

8. The operating system

Managing resources	Describe the need for and the role of the operating system kernel in managing resources, including peripherals, processes, memory protection and backing store.
Providing an interface	Describe the need for and the role of the operating system in providing an interface between the user and the hardware.
Managing backing store	Explain the hierarchical structure of a directory and describe file attributes.
Utility software	Explain the need for and use of a range of utility software.

Modes of operation	Describe the main features of batch processing, real time control and real time transaction systems.
	Identify and describe applications that would be suitable to these modes of operation.
Consideration of human-computer interaction	Explain the need to design systems that are appropriate to the variety of different users at all levels and in different environments.

9. Algorithms and programs

	Explain the term algorithm and describe common methods of defining algorithms, including pseudo-code, flowcharts and structured English.
Variables and constants	Identify and explain the use of constants and variables in algorithms and programs.
Identifiers	Describe why the use of self-documenting identifiers, annotation and program layout are important in programs.
	Give examples of self-documenting identifiers, annotation and appropriate program layout.
Scope of variables	Describe the scope and lifetime of variables in algorithms and programs.
Parameters	Explain the purpose and effect of procedure calling, parameter passing and return, call by reference and call by value.
Mathematical operations	Identify, explain and use mathematical operations in algorithms, including DIV and MOD.
Sorting	Describe the characteristics of sorting algorithms: bubble sort and insertion sort.
Searching	Explain and apply a linear search algorithm.
	Explain and apply the binary search algorithm.
	Describe appropriate circumstances for the use of each searching technique.
	Follow search and sort algorithms and programs and make alterations to such algorithms.
	Write search algorithms and programs.
Problem analysis	Analyse a problem using appropriate design approaches.

Programming constructs	Identify, explain and use sequence, selection and repetition in algorithms and programs.
	Identify, explain and use counts and rogue values in algorithms and programs.
	Follow algorithms and programs involving sequence, selection and repetition and make alterations to such algorithms.
	Write algorithms and programs involving sequence, selection and repetition to solve non-standard problems.
Modular programming	Identify and explain the nature, use and possible benefits of standard functions, standard modules and user defined subprograms.
Logical operations in algorithms and programs	Identify, use and explain the logical operators AND, OR, NOT and XOR in algorithms and programs.
Compression	Explain data compression and how data compression algorithms are used.
Testing	Select appropriate test data to dry-run a program or algorithm in order to identify possible errors.
	Explain the purpose of a given algorithm by showing the effects of test data.

10. Principles of programming

	Describe the distinguishing features of different types of programming paradigms, including procedural, event-driven, visual and mark-up languages.
Levels of computer language	Describe the differences between high-level and low-level languages.
	Describe the role of an object-oriented approach to programming and the relationship between object, class and method.
	Identify and describe situations that require the use of a high-level or a low-level language.
Types of computer language	Identify and justify which type of language would be best suited to develop a solution to a given problem.

11. Systems analysis

Feasibility	Describe the purpose of a feasibility study and describe the processes that an analyst would carry out during a feasibility study. Explain that proposed solutions must be cost effective, developed to an agreed time scale and within an agreed budget.
Investigation and Analysis	Describe the different methods of investigation. Analysis of a problem using appropriate techniques of abstraction and decomposition.
Design	Represent and interpret systems in an appropriate diagrammatic form showing the flow of data and the information processing requirements. Describe the selection of suitable software and hardware to address the requirements of a problem.
Changeover	Describe the various methods of changeover: direct, pilot, phased and parallel, identify the most suitable in a given situation and their relative merits.
Program testing	Describe the use of alpha, beta and acceptance testing.
Maintenance	Describe the nature and use made of perfective, adaptive and corrective maintenance.
Backup and recovery	Describe different procedures for backing up and recovering data.
Documentation	Describe the contents and use made of user documentation and maintenance documentation, including annotated listings, variable lists, algorithms and data dictionaries.

12. Software engineering

Software tools	Explain the role of Integrated Development Environment (IDE) tools in developing and debugging programs.

13. Program construction

Compilers, interpreters and assemblers	Describe the principal stages involved in the compilation process: lexical analysis, symbol table construction, syntax analysis, semantic analysis, code generation and optimisation.

14. The need for different types of software systems and their attributes

Types of software	Explain the use of a range of types of software, including open source software, bespoke and off-the-shelf.
Industrial, technical and scientific	Describe the role of the computer in weather forecasting, computer aided design, robotics and the use of computer generated graphics and animation.
Expert systems	Explain the purpose, use and significance of expert systems.

15. Practical programming

Designing programs	Design all the required data structures to a given problem and fully define them in terms of fieldnames, data types, key fields and requirements for validation. Consider and fully describe the methods of access. Design programming routines to be used to handle and process data for a given problem. Document these designs using a structured convention such as pseudo-code.
Develop programs	Use a documented design to produce a functional prototype to a given problem. Demonstrate understanding of code used by producing annotated listings.
Evaluate programs	Identify the successful features of a system and make specific suggestions for improving less successful areas of the system.

16. Data security and integrity processes

Privacy and security	Describe the dangers that can arise from the use of computers to manage files of personal data. Describe contemporary processes that protect the security and integrity of data including standard clerical procedures, levels of permitted access, passwords for access and write-protect mechanisms.
Disaster planning	Describe the various potential threats to computer systems. Describe contingency planning to recover from disasters.
Malicious and accidental damage	Describe malicious and accidental damage to data and identify situations where either could occur.

17. Economic, moral, legal, ethical and cultural issues relating to computer science

	Describe social and economic changes occurring as a result of developments in computing and computer use, and their moral, ethical, legal, cultural and other consequences.
Effect on employment	Describe the possible effects of computers on the nature of employment in the computing industry and wider society.
Legislation	Explain how relevant legislation impacts on security, privacy, data protection and freedom of information.

3 ASSESSMENT

3.1 Assessment objectives and weightings

Below are the assessment objectives for this specification. Learners must demonstrate their ability to:

AO1
Demonstrate knowledge and understanding of the principles and concepts of computer science, including abstraction, logic, algorithms and data representation

AO2
Apply knowledge and understanding of the principles and concepts of computer science, including to analyse problems in computational terms

AO3
Design, program and evaluate computer systems that solve problems, making reasoned judgements about these and presenting conclusions

The table below shows the weighting of each assessment objective for each component and for the qualification as a whole.

	AO1	AO2	AO3
Component 1	40%	25%	5%
Component 2	-	10%	20%
Total	**40%**	**35%**	**25%**

3.2 Arrangements for Component 2 on-screen examination

This assessment will be carried out in accordance with the instructions set out in 'Instructions for conducting on-screen tests', Appendix 1 of *General, Vocational and Diploma Qualifications: Instructions for conducting examinations* (Joint Council for Qualifications). This document is available on the JCQ website (www.jcq.org.uk).

Programming languages for Component 2

WJEC will support the following programming languages:

- Python
- Visual Basic.NET
- Java

Some tasks in Component 2 will require work to be completed using Python, Java or Visual Basic.NET which have integrated development environments (IDE) freely available for legal download.

Centres will be required to inform WJEC of their choice of language and IDE at the start of the course.

WJEC will supply a paper copy of the assessment tasks and a file/files for each candidate.

Candidates will need access to a computer with:

- A 'clean' user area or storage device on which to save their work
- No access to the Internet or email
- Access to a word processor or similar software to produce their responses
- A functional copy of an IDE for their chosen programming language pre-installed
- File(s) supplied by WJEC that have been pre-installed and tested for use in the assessment.

The practical assessment should be carried out under formal supervision, i.e. the candidates must be in direct sight of the supervisor at all times. Use of resources is tightly prescribed and interaction with other candidates is forbidden.

At the end of the assessment candidate work must be copied to a secondary storage medium. Each candidate's work should be saved in a separate folder labelled with the centre number, candidate number and the first two initials of the candidate's surname and the first initial of the candidate's first name. For example Diane Smith (centre number 68999, candidate number 12345) would store her work in a folder named 68999_12345_SM_D.

The secondary storage medium should be sent for marking to an address supplied by WJEC. The centre must also keep an electronic copy of the candidate's work in a secure location in case of loss or damage to the original submission.

4 TECHNICAL INFORMATION

4.1 Making entries

This is a linear qualification in which all assessments must be taken at the end of the course. Assessment opportunities will be available in the summer series each year, until the end of the life of this specification. Summer 2016 will be the first assessment opportunity.

A qualification may be taken more than once. Candidates must resit all examination components in the same series.

The entry code appears below.

WJEC Eduqas AS Computer Science B500QS

The current edition of our *Entry Procedures and Coding Information* gives up-to-date entry procedures.

4.2 Grading, awarding and reporting

Scaling factors are applied to marks in order for them to achieve their intended weightings. In the case of WJEC Eduqas GCE AS in Computer Science a scaling factor of 1.4 is applied to Component 1, with Component 2 remaining unscaled.

Component	Maximum raw mark	Scaling factor	Scaled maximum mark	% weighting
Component 1	100	1.4	140	70
Component 2	60	1	60	30

AS qualifications are reported as a grade on the scale from A to E. Results not attaining the minimum standard for the award will be reported as U (unclassified).

AS qualifications are free-standing and are awarded in their own right. Assessments at AS cannot contribute to an A level grade.

APPENDIX A - Mathematical skills

Computer science uses mathematics to express its computational laws and processes.

This specification in computer science contains a minimum of 10% mathematics as required by the regulator. Students are asked to demonstrate their knowledge and understanding and skills of computational processes and problem-solving in both theoretical and practical ways. The following list shows the key concepts that are common to all specifications in computer science.

For each topic below, while the concepts are Level 2 (though not all appear in GCSE Mathematics specifications), students will, however, be expected to apply the skills they acquire in a Level 3 context.

Topics:

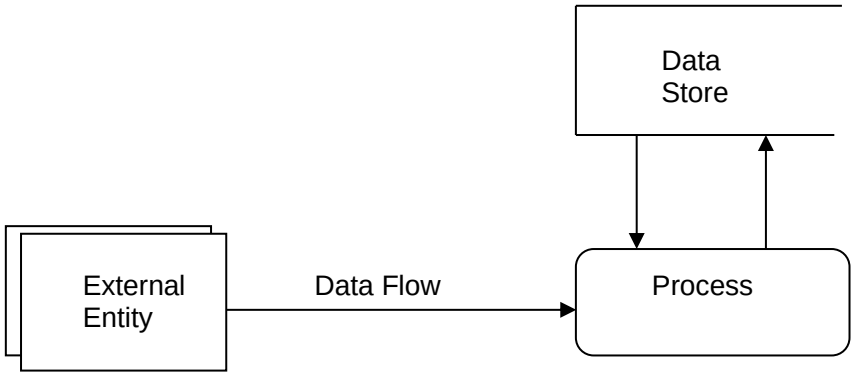
- Boolean algebra
- Number representations and bases

APPENDIX B - Conventions followed in specification

Note 1

In general, the diagram conventions used will be those described in the current edition of *'The BCS Glossary of ICT and Computing Terms'* (published BCS Learning and Development Ltd).

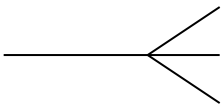
In the case of data flow diagrams, where no generally accepted symbols currently exist, candidates should be familiar with the following symbols, used in a number of current GCE textbooks:

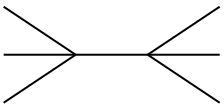


Note 2

The following symbols are used for entities and relationships.

Entity 

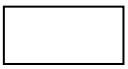
One-To-Many Relationship 

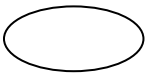
Many-To-Many Relationship 

Note 3

Where Von Neumann architecture is represented diagrammatically, the following symbols are used:

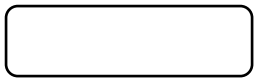
Arithmetic logic unit 

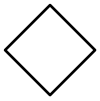
Register 

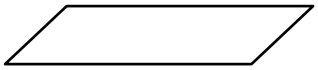
Control unit 

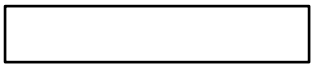
Note 4

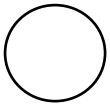
The following symbols are used in flowcharts

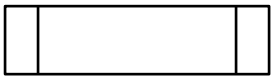
Start / Stop procedure 

Decision box 

Input / Output 

Operation 

Connector 

Store / Subroutine call 

Flow of control