AS and A LEVEL
COMPUTER SCIENC

Year 12 - AS Level	Terms
Component 01: Computing Principles	
Structure and Function of Processor	HT1
Types of Processor	HT1
Input, Output and storage	HT1
Operating Systems	HT2
Applications Generation	HT6
Introduction to Programing	HT6
Databases	HT5
Networks	HT2
Web Technologies	HT2
Data Types	HT3
Data Structures	HT3
Boolean Algebra	HT4
Computing Related Legislation	HT5
Ethic, moral and cultural issues	HT5
Component 02: Algorithms and Problem Solving	
Thinking Abstractly	HT1
Thinking Ahead	HT1
Thinking Procedurally (revisit with Programming Tech).	HT1
Thinking Logically	HT1
Algorithms (revisit with programming Tech).	HT2
Programming Techniques	HT3/ HT4
Software Development	HT5/ HT6
Start on Component 03	HT6 - 6 lessons
Problem identification	
Stakeholders	

Please note that we have Subject Clarification Documents which furth
taught. We would recommend that topics are taught to encourage a breadth

A Level Content Clarification Document
AS Level Content Clarification Document

Year 13 - A Level	Terms
Component 1: Computer Systems	
Structure and Function of Processor	HT3
Types of Processor	HT3
Input, Output and storage	HT4
Systems Software	HT2 - Re-Visit
Software Development	HT2 - Re-Visit
Types of Programming Language	HT2 - Re-Visit
Compression, Encryption and Hashing	HT4
Databases	HT1
Networks	HT4
Web Technologies	HT2 - Re-Visit
Data Types	HT1 - A2 Content
Data Structures	HT3
Boolean Algebra	HT1 - A2 Content
Computing Related Legislation	HT1
Ethic, moral and cultural issues	HT1
Component 02: Algorithms and Problem Solving	
Thinking Abstractly	HT1 - Re-visit
Thinking Ahead	HT1 - Re-visit
Thinking Procedurally with Programming Tech.	HT2 - Re-Visit
Thinking Logically	HT1 - Re-visit
Thinking Concurrently	HT1
Programming Techniques	HT2
Algorithms	HT2 & HT3
Computation Methods	HT4 & HT5

er break down the specification content. Reference should be made to thes
ı of understanding for students, rather than explicitly limit teaching to meet tl

Year 13 - A Level	Terms
Component 03: Programming Project	
Analysis of the problem (10 marks)	HT1
Problem identification	Year 12 - HT6
Stakeholders	Year 12 - HT6
Research the problem	
Specify the proposed solution	
Design of the solution (15 marks)	HT1/ HT2
Decompose the problem	
Describe the solution	
Describe the approach to testing	
Developing the solution (25 marks)	
Iterative development process	HT2
Testing to inform development	HT3/ HT4
Evaluation (20 marks)	HT4
Testing to inform evaluation	
Success of the solution	
Describe the final product	
Maintenance and development	
Programming Project to be finalised before 20th April.	

e, where a centre seeks further guidance as to the level of depth to which content should be
he minimum requirements of the specification.

AS and A LEVEL
COMPUTER SC

Scheme of work teaching hours	
Teaching weeks:	35
Teaching hours/week:	5
Total teaching time:	175
Scheme teaching time:	**121**

Please read guidance notes on right when planning your scheme of work

Teaching Hours	Topic
6	Structure and Function of Processor
9	Types of Processor
8	Input, Output and storage
9	Systems Software (A Level) Operating Systems (AS

	Operating Systems (AS Level)
8	Applications Generation
8	Software Development
20	Types of Programming Language
7	Compression, Encryption and Hashing
6	Databases

5	Networks
7	Web Technologies
8	Data Types
10	Boolean Algebra
5	Computing related legislation
5	Moreal and Ethical Issues

Guidance to teachers

Allocated teaching time is assuming that you are teaching the full A Level, and thus need to include the Component 03 Programming Project.

If solely teaching AS Level, you can adjust teaching time accordingly, as you will not need to factor in the time for the Programming Project, but will need to deliver the Component 02 content.

AS Level is viewed as a ***one year course*** *and fits well with A Level if co-teaching. Stretching AS over two years may make co-teaching significantly more difficult to coordinate.*

Centres may wish to deliver AS in Year 12 and A Level in Year 13. Candidates likely to move on to Year 13 should be studying programming throughout the first year to consolidate their skills. Solely delivering practical programming skills in Year 13 may hinder their progress within the Component 03 Programming Project unit.

Sub Topic
The Arithmetic and Logic Unit; ALU, Control Unit and Registers (Program Counter; PC, Accumulator; ACC, Memory Address Register; MAR, Memory Data Register; MDR, Current Instruction Register; CIR). Buses: data, address and control
The Fetch-Decode-Execute Cycle; including its effects on registers.
The factors affecting the performance of the CPU: clock speed, number of cores, cache.
The use of pipelining in a processor to improve efficiency
Von Neumann, Harvard and contemporary processor architecture.
The differences between and uses of CISC and RISC processors.
GPUs and their uses (including those not related to graphics).
Multicore and Parallel systems.
How different input, output and storage devices can be applied to the solution of different problems.
The uses of magnetic, flash and optical storage devices.
RAM and ROM.
Virtual storage.
The need for, function and purpose of operating systems.
Memory Management (paging, segmentation and virtual memory).
Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch-Decode-Execute Cycle.
Scheduling: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time.

Distributed, embedded, multi-tasking, multi-user and Real Time operating systems.

BIOS.

Device drivers.

Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another.

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

Utilities.

Open source vs. closed source.

Translators: Interpreters, compilers and assemblers.

Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation).

Linkers and loaders and use of libraries.

Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.

The relative merits and drawbacks of different methodologies and when they might be used.

Writing and following algorithms.

Different test strategies, including black and white box testing and alpha and beta testing

Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.

Need for and characteristics of a variety of programming paradigms.

Procedural languages:

- program flow
- variables and constants
- procedures and functions
- arithmetic, Boolean and assignment operators
- string handling
- file handling.

Assembly language (including following and writing simple programs with the Little Man Computer instruction set).

Modes of addressing memory (immediate, direct, indirect and indexed).

Object-oriented languages with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism.

Lossy vs. Lossless compression.

Run length encoding and dictionary coding for lossless compression.

Symmetric and asymmetric encryption.

Different uses of hashing.

Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling, normalisation and indexing.

Methods of capturing, selecting, managing and exchanging data.

Normalisation to 3NF.

SQL – Interpret and modify.

Referential integrity.

Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy.

Characteristics of networks and the importance of protocols and standards.

The internet structure:

- The TCP/IP Stack.
- DNS
- Protocol layering.
- LANs and WANs.
- Packet and circuit switching.

Network security and threats, use of firewalls, proxies and encryption.
Network hardware.
Client-server and peer to peer.

HTML, CSS and JavaScript.
Search engine indexing.
PageRank algorithm.
Server and client side processing.

Primitive data types, integer, real/floating point, character, string and Boolean.
Represent positive integers in binary.
Use of sign and magnitude and two’s complement to represent negative numbers in binary.
Addition and subtraction of binary integers.
Represent positive integers in hexadecimal.
Convert positive integers between binary hexadecimal and denary.
Representation and normalisation of floating point numbers in binary.
Floating point arithmetic, positive and negative numbers, addition and subtraction.
Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR.
Positive and negative real numbers using normalised floating point representation
How character sets (ASCII and UNICODE) are used to represent text.

Define problems using boolean logic.
Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions
Use the following rules to derive or simplify statements in Boolean algebra: De Morgan’s Laws, distribution, association, commutation, double negation.
Using logic gate diagrams and truth tables.
The logic associated with D type flip flops, half and full adders.

The Data Protection Act 1998.
The Computer Misuse Act 1990.
The Copyright Design and Patents Act 1988.
The Regulation of Investigatory Powers Act 2000.

The individual moral, social, ethical and cultural opportunities and risks of digital technology:

- Computers in the workforce.
- Automated decision making.
- Artificial intelligence.
- Environmental effects.
- Censorship and the Internet.
- Monitor behaviour.
- Analyse personal information.
- Piracy and offensive communications.
- Layout, colour paradigms and character sets.

Common Content	AS only Content	A Level Content	Co-teachable percentage
🟩			80%
🟩			
🟩			
		🟦	
🟩			
🟩			66%
		🟦	
🟩			
🟩			100%
🟩			
🟩			
🟩			
🟩			100%
🟩			
🟩			
🟩			

			83%
			60%
			40%
			25%
			33%

			60%
			25%
			64%
			60%
			100%
			100%

Resource Links
Online Delivery Guide: Structure and function of the processor Topic Exploration Pack - Teacher Instructions: Structure and function of the Processor Learner Activity: Structure and Function of the Processor
Online Delivery Guide: Types of Processor Topic Exploration Pack - Teacher Instructions: Types of Processor Topic Exploration Pack - Learner Activity: Types of Processor
Online Delivery Guide: Input, output, storage
Online Delivery Guide: Systems Software Topic Exploration Pack - Teacher Instructions: Systems Software Topic Exploration Pack - Learner Activity: Systems Software

Topic Exploration Pack - Teacher Instructions: Application Generation Topic Exploration Pack - Learner Activity: Application Generation
Online Delivery Guide: Software Development Topic Exploration Pack - Teacher Instructions: Software Development Topic Exploration Pack - Learner Activity: Software Development
Online Delivery Guide: Types of Programming Language Topic Exploration Pack - Learner Activity: Types of Programming Language
Online Delivery Guide: Compression, Encryption and Hashing Topic Exploration Pack - Teacher Instructions: Compression, Encryption and Hashing *For AS LEVEL Lossy v Lossless Compression is part of 1.3.3 Web Technologies*
Online Delivery Guide: Software Development Topic Exploration Pack - Teacher Instructions: Software Development Topic Exploration Pack - Learner Activity: Software Development
Online Delivery Guide: Networks

Topic Exploration Pack - Teacher Instructions: Networks
Online Delivery Guide: Web Technologies Topic Exploration Pack - Teacher Instructions: Web Technologies Topic Exploration Pack - Learner Activity: Web Technologies
Online Delivery Guide: Data Types Topic Exploration Pack - Teacher Instructions: Data Types Topic Exploration Pack - Learner Activity 1: Data Types Topic Exploration Pack - Learner Activity 2: Data Types Topic Exploration Pack - Learner Activity 3: Data Types
Online Delivery Guide: Boolean Algebra Topic Exploration Pack - Teacher Instructions: Boolean Algebra Topic Exploration Pack - Learner Activity 1: Boolean Algebra Topic Exploration Pack - Learner Activity 2: Boolean Algebra Topic Exploration Pack - Learner Activity 3: Boolean Algebra
No current supporting resources for this unit
No current supporting resources for this unit

AS and A LEVEL
COMPUTER SC

Scheme of work teaching hours	
Teaching weeks:	30
Teaching hours/week:	5
Total teaching time:	150
Scheme teaching time:	**90**

Please read guidance notes on right when planning your scheme of work

Teaching Hours	Topic
5	Thinking Abstractly
5	Thinking Ahead
5	Thinking Procedurally
5	Thinking Logically
5	Thinking Concurrently

30	Programming Techniques
10	Software Development
10	Computational Methods
15	Algorithms

IENCE

Guidance to teachers

Allocated teaching time is assuming that you are teaching the full A Level, and thus need to include the Component 03 Programming Project.

If solely teaching AS Level then you can adjust teaching time accordingly, as you will not need to factor in the time for the Programming Project, but will need to deliver the Component 02 content.

AS Level is viewed as a ***one year course*** *and fits well with A Level if co-teaching. Stretching AS over two years may make co-teaching significantly more difficult to coordinate.*

Centres may wish to deliver AS in Year 12 and A Level in Year 13. Candidates likely to move on to Year 13 should be studying programming throughout the first year to consolidate their skills. Solely delivering practical programming skills in Year 13 may hinder their progress within the Component 03 Programming Project unit.

Sub Topic	Common Content	AS only Content	A Level Content
The nature of abstraction.	🟩		
The need for abstraction.	🟩		
The differences between an abstraction and reality.	🟩		
Devise an abstract model for a variety of situations.	🟩		
Identify the inputs and outputs for a given situation.	🟩		
Determine the preconditions for devising a solution to a problem.	🟩		
The nature, benefits and drawbacks of caching.			🟦
The need for reusable program components.	🟩		
Identify the components of a problem.	🟩		
Identify the components of a solution to a problem.	🟩		
Determine the order of the steps needed to solve a problem.	🟩		
Identify sub-procedures necessary to solve a problem.	🟩		
Identify the points in a solution where a decision has to be taken.	🟩		
Determine the logical conditions that affect the outcome of a decision.	🟩		
Determine how decisions affect flow through a program.	🟩		
Determine the parts of a problem that can be tackled at the same time.			🟦
Outline the benefits and trade offs that might result from concurrent processing in a particular situation.			🟦
Programming constructs: sequence, iteration, branching.	🟩		

Recursion, how it can be used and compares to an iterative approach.			■
Global and local variables.	■		
Modularity, functions and procedures, parameter passing by value and by reference.	■		
Use of an IDE to develop/debug a program.	■		
Use of object oriented techniques.			■
Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.		■	
The relative merits and drawbacks of different methodologies and when they might be used.		■	
Writing and following algorithms.		■	
Different test strategies, including black and white box testing and alpha and beta testing		■	
Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.		■	
Features that make a problem solvable by computational methods.			■
Problem recognition.			■
Problem decomposition.			■
Use of divide and conquer.			■
Use of abstraction.			■
Learners should apply their knowledge of: • backtracking • data mining • heuristics • performance modelling • pipelining • visualisation to solve problems.			■
Analysis and design of algorithms for a given situation.	■		
The suitability of different algorithms for a given task and data set, in terms of execution time and space.			■
Standard algorithms (bubble sort, insertion sort, binary search and linear search).	■		
Standard algorithms (quick sort, Dijkstra’s shortest path algorithm, A* algorithm, binary search).			■
Implement bubble sort, insertion sort.	■		
Implement binary and linear search.	■		
Representing, adding data to and removing data from queues and stacks.		■	
Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity).			■
Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees).			■
Comparison of the complexity of algorithms.			■
Compare the suitability of different algorithms for a given task and data set.		■	

Co-teachable percentage	Resource Links
100%	Online Delivery Guide: Thinking Abstractly
75%	Online Delivery Guide: Thinking Ahead Topic Exploration Pack - Teacher Instructions: Thinking Ahead Topic Exploration Pack - Learner Activity: Thinking Ahead
100%	*No current supporting resources for this unit*
100%	Topic Exploration Pack - Teacher Instructions: Thinking Logically Topic Exploration Pack - Learner Activity: Thinking Logically
0%	Online Delivery Guide: Thinking Concurrently Topic Exploration Pack - Teacher Instructions: Thinking Concurrently Topic Exploration Pack - Learner Activity: Thinking Concurrently Topic Exploration Pack - Learner Activity: Thinking Concurrently
	Online Delivery Guide: Programming Techniques

66%	Topic Exploration Pack - Teacher Instructions: Programming Techniques Topic Exploration Pack - Learner Activity: Programming Techniques
0%	Online Delivery Guide: Software Development Topic Exploration Pack - Teacher Instructions: Software Development Topic Exploration Pack - Learner Activity: Software Development
0%	Online Delivery Guide: Computational Methods Topic Exploration Pack - Teacher Instructions: Computational Methods
37%	Topic Exploration Pack - Teacher Instructions: Algorithms Topic Exploration Pack - Learner Activity: Algorithms Topic Exploration Pack - Learner Activity: Activity 2 Program Code Topic Exploration Pack - Learner Activity: Activity 4 Program Code

AS and A LEVEL
COMPUTER SC

Scheme of work teaching hours	
Suggested teaching time	70-80

Please read guidance notes on right when planning your scheme of work

Teaching Hours	Topic
Analysis of the Problem (10 Marks)	
11	Problem Identification
	Stakeholders
	Research the Problem
	Specify the Proposed Solution
Design of the solution (15 Marks)	
	Decompose the Problem
	Describe the solution

17	Describe the solution
	Describe the approach to testing
Developing the solution (25 Marks)	
35	Iterative Development Process Testing to inform development
Evaluation (20 Marks)	
15	Testing to inform evaluation
	Success of the solution
	Describe the final product
	Maintenance and development

Guidance to teachers

Component 03 (Programming Project) is a personal programming project, driven by the student. There is no limit as to how much time you may spend on it - however, candidates will need to have developed understanding of procedural and object-oriented programming, as well as sorting and searching algorithms and the advanced data structures.

The programming project is worth 20% of the final mark.

The project is expected to be around 20-25% of the total teaching time - although realistically it may take slightly more. Most schools tend to commence idea generation and project idea submission towards the end of Year 12 (1st Year of A Level) in preparation for the following September.

Whilst there are not limits on how long to spend on each section, the teaching hours are based on the percentage of marks allocated to that section. Teachers should use their discretion and judgement when deciding teaching hours.

Sub Topic
Describe and justify the features that make the problem solvable by computational methods.
Explain why the problem is amenable to a computational approach.
Identify and describe those who will have an interest in the solution explaining how the solution is appropriate to their needs (this may be named individuals, groups or persona that describes the target end user).
Research the problem and solutions to similar problems to identify and justify suitable approaches to a solution.
Describe the essential features of a computational solution explaining these choices.
Explain the limitations of the proposed solution.
Identify the points in a solution where a decision has to be taken.
Determine the logical conditions that affect the outcome of a decision
Determine how decisions affect flow through a program.
Break down the problem into smaller parts suitable for computational solutions justifying any decisions made.
Explain and justify the structure of the solution
Describe the parts of the solution using algorithms justifying how these algorithms form a complete solution to the problem.

Describe usability features to be included in the solution.
Identify key variables / data structures / classes justifying choices and any necessary validation.
Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development.
The relative merits and drawbacks of different methodologies and when they might be used.
Writing and following algorithms.
Different test strategies, including black and white box testing and alpha and beta testing.
Test programs that solve problems using suitable test data and end user feedback, justify a test strategy for a given situation.
Provide annotated evidence of each stage of the iterative development process justifying any decision made. Provide annotated evidence of prototype solutions justifying any decision made. Provide annotated evidence for testing at each stage justifying the reason for the test. Provide annotated evidence of any remedial actions taken justifying the decision made.
Provide annotated evidence of testing the solution of robustness at the end of the development process. Provide annotated evidence of usability testing (user feedback).
Use the test evidence from the development and post development process to evaluate the solution against the success criteria from the analysis.
Provide annotated evidence of the usability features from the design, commenting on their effectiveness.
Discuss the maintainability of the solution. Discuss potential further development of the solution.

Resource Links

CPD Courses for delivery of Programming Project

Centre Authentication and Candidate Record Forms

Centre Authentication Form

Interactive Unit Record Sheet

Other Supporting resources

Types of Programming Languages

Data Types Delivery Guide

Data Structures Delivery Guide

Programming Techniques

Thinking Abstractly

Thinking Ahead

Thinking Concurrently

Teacher Guides

Project teacher guide

Programming Language Guide

Project Complexity Guide

Pseudocode Guide