## 1.1 Systems Architecture

### 1.1.1 Architecture of the CPU

- The purpose of the CPU:
  - The fetch-execute cycle
  - Common CPU components and their function: ALU (Arithmetic Logic Unit)
  - CU (Control Unit)
  - Cache
  - Registers
- Von Neumann architecture:
  - MAR (Memory Address Register)
  - MDR (Memory Data Register)
  - Program Counter
  - Accumulator

Required
- What actions occur at each stage of the fetch-execute cycle
- The role/purpose of each component and what it manages, stores, or controls during the fetch-execute cycle
- The purpose of each register, what it stores (data or address)
- The difference between storing data and an address

Not required
- Knowledge of passing of data between registers in each stage

### 1.1.2 CPU performance

- How common characteristics of CPUs affect their performance:
  - Clock speed
  - Cache size
  - Number of cores

Required
- Understanding of each characteristic as listed
- The effects of changing any of the common characteristics on system performance, either individually or in combination

### 1.1.3 Embedded systems

- The purpose and characteristics of embedded systems
- Examples of embedded systems

Required
- What embedded systems are
- Typical characteristics of embedded systems
- Familiarity with a range of different embedded systems
-

## 1.2 – Memory and storage

### 1.2.1 Primary storage (Memory)

- The need for primary storage
- The difference between RAM and ROM
- The purpose of ROM in a computer system
- The purpose of RAM in a computer system
- Virtual memory

Required
- Why computers have primary storage
  - How this usually consists of RAM and ROM
- Key characteristics of RAM and ROM
- Why virtual memory may be needed in a system
- How virtual memory works
  - Transfer of data between RAM and HDD when RAM is filled

<table>
<tr><th colspan="2">1.2.2 Secondary storage</th></tr>
<tr>
<td>

- The need for secondary storage
- Common types of storage:
  - Optical
  - Magnetic
  - Solid state
- Suitable storage devices and storage media for a given application
- The advantages and disadvantages of different storage devices and storage media relating to these characteristics:
  - Capacity
  - Speed
  - Portability
  - Durability
  - Reliability
  - Cost

</td>
<td>

Required

- Why computers have secondary storage
- Recognise a range of secondary storage devices/media
- Differences between each type of storage device/medium
- Compare advantages/disadvantages for each storage device
- Be able to apply their knowledge in context within scenarios

Not required

- Understanding of the component parts of these types of storage

</td>
</tr>
<tr><th colspan="2">1.2.3 Units</th></tr>
<tr>
<td>

- The units of data storage:
  - Bit
  - Nibble (4 bits)
  - Byte (8 bits)
  - Kilobyte (1,000 bytes or 1 KB)
  - Megabyte (1,000 KB)
  - Gigabyte (1,000 MB)
  - Terabyte (1,000 GB)
  - Petabyte (1,000 TB)
- How data needs to be converted into a binary format to be processed by a computer
- Data capacity and calculation of data capacity requirements

</td>
<td>

Required

- Why data must be stored in binary format
- Familiarity with data units and moving between each
- Calculate capacity of devices
- Calculate required capacity for a given set of files
- Calculate file sizes of sound, images and text files
  - sound file size = sample rate x duration (s) x bit depth
  - image file size = colour depth x image height (px) x image width (px)
  - text file size = bits per character x number of characters

Alternatives

- Use of 1,024 for conversions and calculations would be acceptable
- Allowance for metadata in calculations may be used

</td>
</tr>
<tr><th colspan="2">1.2.4 Data storage</th></tr>
<tr>
<td>

**Numbers**

- How to convert positive denary whole numbers to binary numbers (up to and including 8 bits) and vice versa
- How to add two binary integers together (up to and including 8 bits) and explain overflow errors which may occur

</td>
<td>

Required

- Denary number range 0 – 255
- Hexadecimal range 00 – FF
- Binary number range 00000000 – 11111111
- Understanding of the terms most significant bit, and least significant bit

</td>
</tr>
</table>

| | |
|---|---|
| ▪ How to convert positive denary whole numbers into 2-digit hexadecimal numbers and vice versa<br>▪ How to convert binary integers to their hexadecimal equivalents and vice versa<br>▪ Binary shifts<br><br>**Characters**<br>▪ The use of binary codes to represent characters<br>▪ The term 'character set'<br>▪ The relationship between the number of bits per character in a character set, and the number of characters which can be represented, e.g.:<br>o ASCII<br>o Unicode<br><br>**Images**<br>▪ How an image is represented as a series of pixels, represented in binary<br>▪ Metadata<br>▪ The effect of colour depth and resolution on:<br>o The quality of the image<br>o The size of an image file<br>**Sound**<br>▪ How sound can be sampled and stored in digital form<br>▪ The effect of sample rate, duration and bit depth on:<br>o The playback quality<br>o The size of a sound file | ▪ Conversion of any number in these ranges to another number base<br>▪ Ability to deal with binary numbers containing between 1 and 8 bits<br>o e.g. 11010 is the same as 00011010<br>▪ Understand the effect of a binary shift (both left or right) on a number<br><br>Required<br>▪ How characters are represented in binary<br>▪ How the number of characters stored is limited by the bits available<br>▪ The differences between and impact of each character set<br>▪ Understand how character sets are logically ordered, e.g. the code for 'B' will be one more than the code for 'A'<br>▪ Binary representation of ASCII in the exam will use 8 bits<br>Not required<br>▪ Memorisation of character set codes<br><br>Required<br>▪ Each pixel has a specific colour, represented by a specific code<br>▪ The effect on image size and quality when changing colour depth and resolution<br>▪ Metadata stores additional image information (e.g. height, width, etc.)<br><br>Required<br>▪ Analogue sounds must be stored in binary<br>▪ Sample rate – measured in Hertz (Hz)<br>▪ Duration – how many seconds of audio the sound file contains<br>▪ Bit depth – number of bits available to store each sample (e.g. 16-bit) |
| **1.2.5 Compression** | |
| ▪ The need for compression<br>▪ Types of compression:<br>o Lossy<br>o Lossless | Required<br>▪ Common scenarios where compression may be needed<br>▪ Advantages and disadvantages of each type of compression<br>▪ Effects on the file for each type of compression<br><br>Not required<br>▪ Ability to carry out specific compression algorithms |

# 1.3 – Computer networks, connections and protocols

## 1.3.1 Networks and topologies

- Types of network:
  - LAN (Local Area Network)
  - WAN (Wide Area Network)
- Factors that affect the performance of networks
- The different roles of computers in a client-server and a peer-to-peer network
- The hardware needed to connect stand-alone computers into a Local Area Network:
  - Wireless access points
  - Routers
  - Switches
  - NIC (Network Interface Controller/Card)
  - Transmission media
- The Internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - Hosting
  - The Cloud
  - Web servers and clients
- Star and Mesh network topologies

Required

- The characteristics of LANs and WANs including common examples of each
- Understanding of different factors that can affect the performance of a network, e.g.:
  - Number of devices connected
  - Bandwidth
- The tasks performed by each piece of hardware
- The concept of the Internet as a network of computer networks
- A DNS's role in the conversion of a URL to an IP address
- Concept of servers providing services (e.g. Web server " Web pages, File server " file storage/retrieval)
- Concept of clients requesting/using services from a server
- The Cloud: remote service provision (e.g. storage, software, processing)
- Advantages and disadvantages of the Cloud
- Advantages and disadvantages of the Star and Mesh topologies
- Apply understanding of networks to a given scenario

## 1.3.2 Wired and wireless networks, protocols and layers

- Modes of connection:
  - Wired
    - Ethernet
  - Wireless
    - Wi-Fi
    - Bluetooth
- Encryption
- IP addressing and MAC addressing
- Standards
- Common protocols including:
  - TCP/IP (Transmission Control Protocol/Internet Protocol)
  - HTTP (Hyper Text Transfer Protocol)
  - HTTPS (Hyper Text Transfer Protocol Secure)
  - FTP (File Transfer Protocol) o POP (Post Office Protocol)
  - IMAP (Internet Message Access Protocol)
  - SMTP (Simple Mail Transfer Protocol)
- The concept of layers

Required

- Compare benefits and drawbacks of wired versus wireless connection
- Recommend one or more connections for a given scenario
- The principle of encryption to secure data across network connections
- IP addressing and the format of an IP address (IPv4 and IPv6)
- A MAC address is assigned to devices; its use within a network
- The principle of a standard to provide rules for areas of computing
- Standards allows hardware/software to interact across different manufacturers/producers
- The principle of a (communication) protocol as a set of rules for transferring data
- That different types of protocols are used for different purposes
- The basic principles of each protocol i.e. its purpose and key features

| | |
|---|---|
| | ▪ How layers are used in protocols, and the benefits of using layers; for a teaching example, please refer to the 4-layer TCP/IP model<br>Not required<br>▪ Understand how Ethernet, Wi-Fi and Bluetooth protocols work<br>▪ Understand differences between static and dynamic, or public and private IP addresses<br>▪ Knowledge of individual standards<br>▪ Knowledge of the names and function of each TCP/IP layer |
| **1.4 – Network security** | |
| **1.4.1 Threats to computer systems and networks** | |
| ▪ Forms of attack:<br>o Malware<br>o Social engineering, e.g. phishing, people as the 'weak point'<br>o Brute-force attacks<br>o Denial of service attacks<br>o Data interception and theft<br>o The concept of SQL injection | Required<br>▪ Threats posed to devices/systems<br>▪ Knowledge/principles of each form of attack including:<br>▪ How the attack is used<br>▪ The purpose of the attack |
| **1.4.2 Identifying and preventing vulnerabilities** | |
| ▪ Common prevention methods:<br>o Penetration testing<br>o Anti-malware software<br>o Firewalls<br>o User access levels<br>o Passwords<br>o Encryption<br>o Physical security | Required<br>▪ Understanding of how to limit the threats posed in 1.4.1<br>▪ Understanding of methods to remove vulnerabilities<br>▪ Knowledge/principles of each prevention method:<br>o What each prevention method may limit/prevent<br>o How it limits the attack |
| **1.5 – Systems software** | |
| **1.5.1 Operating systems** | |
| ▪ The purpose and functionality of operating systems:<br>o User interface<br>o Memory management and multitasking<br>o Peripheral management and drivers<br>o User management o File management | Required<br>▪ What each function of an operating system does<br>▪ Features of a user interface<br>▪ Memory management, e.g. the transfer of data between memory, and how this allows for multitasking<br>▪ Understand that:<br>o Data is transferred between devices and the processor<br>o This process needs to be managed and what this entails (e.g. the use of buffers when transferring data to a printer) |

| | |
|---|---|
| | ▪ User management functions, e.g.:<br>o Allocation of an account<br>o Access rights<br>o Security, etc.<br>▪ File management, and the key features, e.g.:<br>o Naming<br>o Allocating to folders<br>o Moving files<br>o Saving, etc.<br>Not required<br>▪ Understanding of paging or segmentation |
| **1.5.2 Utility software** | |
| ▪ The purpose and functionality of utility software<br>▪ Utility system software:<br>o Encryption software<br>o Defragmentation<br>o Data compression | Required<br>▪ Understand that computers often come with utility software, and how this performs housekeeping tasks<br>▪ Purpose of the identified utility software and why it is required |
| **1.6 – Ethical, legal, cultural and environmental impacts of digital technology** | |
| **1.6.1 Ethical, legal, cultural and environmental impact** | |
| ▪ Impacts of digital technology on wider society including:<br>o Ethical issues<br>o Legal issues<br>o Cultural issues<br>o Environmental issues<br>o Privacy issues<br>▪ Legislation relevant to Computer Science:<br>o The Data Protection Act 2018<br>o Computer Misuse Act 1990<br>o Copyright Designs and Patents Act 1988<br>o Software licences (i.e. open source and proprietary) | Required<br>▪ Technology introduces ethical, legal, cultural, environmental and privacy issues<br>▪ Knowledge of a variety of examples of digital technology and how this impacts on society<br>▪ An ability to discuss the impact of technology based around the issues listed<br>▪ The purpose of each piece of legislation and the specific actions it allows or prohibits<br>▪ The need to license software and the purpose of a software licence<br>▪ Features of open source (providing access to the source code and the ability to change the software)<br>▪ Features of proprietary (no access to the source code, purchased commonly as off-the-shelf)<br>▪ Recommend a type of licence for a given scenario including benefits and drawbacks |

## 2.1 Algorithms

### 2.1.1 Computational thinking

- Principles of computational thinking:
  - Abstraction
  - Decomposition
  - Algorithmic thinking

**Required**

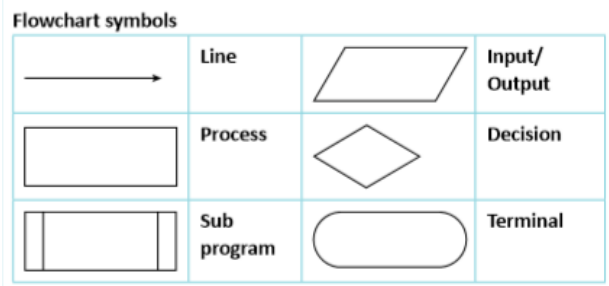
- Understanding of these principles and how they are used to define and refine problems

### 2.1.2 Designing, creating and refining algorithms

- Identify the inputs, processes, and outputs for a problem
- Structure diagrams
- Create, interpret, correct, complete, and refine algorithms using:
  - Pseudocode
  - Flowcharts
  - Reference language/high-level programming language
- Identify common errors
- Trace tables

**Required**

- Produce simple diagrams to show:
  - The structure of a problem
  - Subsections and their links to other subsections
- Complete, write or refine an algorithm using the techniques listed
- Identify syntax/logic errors in code and suggest fixes
- Create and use trace tables to follow an algorithm




### 2.1.3 Searching and sorting algorithms

- Standard searching algorithms:
  - Binary search
  - Linear search
- Standard sorting algorithms:
  - Bubble sort
  - Merge sort
  - Insertion sort

**Required**

- Understand the main steps of each algorithm
- Understand any pre-requisites of an algorithm
- Apply the algorithm to a data set
- Identify an algorithm if given the code for it

**Not required**

- To remember the code for these algorithms

## 2.2 – Programming Fundamentals

### 2.2.1 Programming fundamentals

- The use of variables, constants, operators, inputs, outputs and assignments
- The use of the three basic programming constructs used to control the flow of a program:
  - Sequence
  - Selection
  - Iteration (count-and condition-controlled loops)
- The common arithmetic operators
- The common Boolean operators AND, OR and NOT

**Required**

- Practical use of the techniques in a high-level language within the classroom
- Understanding of each technique
- Recognise and use the following operators:

Comparison Operators:
== Equal to
!= Not equal to
< Less than
<= Less than or equal to
> Greater than
>= Greater than or equal to
Arithmetic Operators:
+ Addition
- Subtraction
* Multiplication
^ Exponent (to the power of)
/ Division
MOD Modulus
DIV Quotient

### 2.2.2 Data types

The use of data types:
Integer
Real
Boolean
Character and string
Casting

**Required**

- Practical use of the data types in a high-level language within the classroom
- Ability to choose suitable data types for data in a given scenario
- Understand that data types may be temporarily changed through casting, and where this may be useful

### 2.2.3 Additional programming techniques

- The use of basic string manipulation
- The use of basic file handling operations:
  - Open
  - Read
  - Write
  - Close
- The use of records to store data
- The use of SQL to search for data
- The use of arrays (or equivalent) when solving problems, including both one-dimensional and two-dimensional arrays
- How to use sub programs (functions and procedures) to produce structured code
- Random number generation

**Required**

- Practical use of the additional programming techniques in a high-level language within the classroom
- Ability to manipulate strings, including:
  - Concatenation
  - Slicing
- Arrays as fixed length static structures
- The use of functions
- The use of procedures
- Where to use functions and procedures effectively
- SQL commands:
  - SELECT
  - FROM
  - WHERE

## 2.3 – Producing Robust Programs

### 2.3.1 Defensive design

| | |
|---|---|
| ▪ Defensive design considerations:<br>  ○ Anticipating misuse<br>  ○ Authentication<br>▪ Input validation<br>▪ Maintainability:<br>  ○ Use of sub programs<br>  ○ Naming conventions<br>  ○ Indentation<br>  ○ Commenting | **Required**<br>▪ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values<br>▪ Understanding of how to deal with invalid data in a program<br>▪ Authentication to confirm the identity of a user<br>▪ Practical experience of designing input validation and simple authentication (e.g. username and password)<br>▪ Understand why commenting is useful and apply this appropriately |

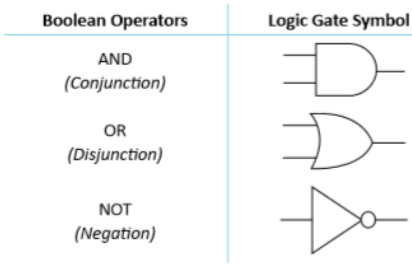
### 2.3.2 Testing

| | |
|---|---|
| ▪ The purpose of testing<br>▪ Types of testing:<br>  ○ Iterative<br>  ○ Final/terminal<br>▪ Identify syntax and logic errors<br>▪ Selecting and using suitable test data:<br>  ○ Normal<br>  ○ Boundary<br>  ○ Invalid<br>  ○ Erroneous<br>▪ Refining algorithms | **Required**<br>▪ The difference between testing modules of a program during development and testing the program at the end of production<br>▪ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated<br>▪ Logic errors as errors which produce unexpected output<br>▪ Normal test data as data which should be accepted by a program without causing errors<br>▪ Boundary test data as data of the correct type which is on the very edge of being valid<br>▪ Invalid test data as data of the correct type but outside accepted validation limit<br>▪ Erroneous test data as data of the incorrect type which should be rejected by a computer system<br>▪ Ability to identify suitable test data for a given scenario<br>▪ Ability to create/complete a test plan |

## 2.4 – Boolean Logic

### 2.4.1 Boolean logic

| | |
|---|---|
| ▪ Simple logicdiagrams using the operators AND, OR and NOT<br>▪ Truth tables<br>▪ Combining Boolean operators using AND, OR and NOT<br>▪ Applying logical operators in truth tables to solve problems | **Required**<br>▪ Knowledge of the truth tables for each logic gate<br>▪ Recognition of each gate symbol<br>▪ Understanding of how to create, complete or edit logic diagrams and truth tables for given scenarios<br>▪ Ability to work with more than one gate in a logic diagram |

| | |
|---|---|
| |  <br>**Truth Tables**<br>AND: A, B, A AND B — 0 0 0; 0 1 0; 1 0 0; 1 1 1<br>OR: A, B, A OR B — 0 0 0; 0 1 1; 1 0 1; 1 1 1<br>NOT: A, NOT A — 0 1; 1 0<br>**Alternatives**<br>• Use of other valid notation will be accepted within the examination, e.g. Using T/F for 1/0, or V for OR, etc. |

## 2.5 – Programming Languages and Integrated Development Environments

### 2.5.1 Languages

| | |
|---|---|
| ▪ Characteristics and purpose of different levels of programming language:<br>  ○ High-level languages<br>  ○ Low-level languages<br>▪ The purpose of translators<br>▪ The characteristics of a compiler and an interpreter | **Required**<br>▪ The differences between high-level and low-level programming languages<br>▪ The need for translators<br>▪ The differences, benefits and drawbacks of using a compiler or an interpreter<br>**Not required**<br>▪ Understanding of assemblers |

### 2.5.2 The Integrated Development Environment (IDE)

| | |
|---|---|
| ▪ Common tools and facilities available in an Integrated Development Environment (IDE):<br>  ○ Editors<br>  ○ Error diagnostics<br>  ○ Run-time environment<br>  ○ Translators | **Required**<br>▪ Knowledge of the tools that an IDE provides<br>▪ How each of the tools and facilities listed can be used to help a programmer develop a program<br>▪ Practical experience of using a range of these tools within at least one IDE |