Cambridge Elevate includes a number of features that enable you to customise this resource:

- You can re-order the contents of this resource through Cambridge Elevate, to suit your teaching sequence.
- You can create your own notes (highlights, annotations, voice notes) to help build information within the resource.
- You can also add hyperlinks between sections within the resource, to sections within other resources within Cambridge Elevate, or to web pages.

You can find more information about the functionality available to you within Cambridge Elevate and how you can use it in your teaching within the Cambridge Elevate Student and Teacher User Guide, which is available from your Home screen.
The Cambridge A/AS Level Computing for WJEC/Eduqas suite of resources covers the full AS and A Level specifications for the Eduqas (England) and WJEC (Wales) qualifications. This outline explains how to use these resources to teach the different pathways.

**A/AS Level Computer Science for WJEC/Eduqas Component 1**

This product is designed to prepare students for written examinations. It covers Component 1 of the Eduqas A Level specification (England) and Component 3 of the WJEC A Level specification (Wales). It also covers Components 1 and 2 of the Eduqas AS Level and WJEC AS/A Level specifications.

**A/AS Level Computer Science for WJEC/Eduqas Component 2**

This product is designed to prepare students for written examinations. It covers Component 2 of the Eduqas A Level specification (England) and Component 4 of the WJEC A Level specification (Wales). It also covers Components 1 and 2 of the Eduqas AS Level and WJEC AS/A Level specifications.

**A/AS Level Computer Science for WJEC/Eduqas Teaching Programming**

This product is designed to help teachers prepare students for on-screen and practical examinations/coursework. It can be used for Component 3 of the Eduqas A Level specification (England) and Component 5 of the WJEC A Level specification (Wales). It can also be used for Component 2 of the Eduqas AS Level and WJEC AS/A Level specifications.

In order to teach A Level in England or Wales, you will need all three of these products.

**AS Level**

The AS Level specifications for England and Wales are identical. In order to teach AS Level in England or Wales, you will need all three of these products. However, the content in these products is presented in the order in which it is studied at A Level in England and A2 in Wales, so you will need to approach it in a slightly different order to teach AS Level.

Here's how we suggest you approach the AS Level content:

1. **Hardware and communication**: Component 2 Chapter 1
2. **Logical operations**: Component 1 Chapter 2
3. **Data transmission**: Component 2 Chapter 2
4. **Data representation and data types**: Component 2 Chapter 3
5. **Data structures**: Component 1 Chapter 1
6. **Organisation and structure of data**: Component 2 Chapter 4
7. **Databases and distributed systems**: Component 2 Chapter 5
8. **The operating system**: Component 2 Chapter 6
9. **Algorithms and programs**: Component 1 Chapter 3
10. **Principles of programming**: Component 1 Chapter 4
11. **Systems analysis**: Component 1 Chapter 5
12. **Software engineering**: Component 1 Chapter 7
13. **Program construction**: Component 2 Chapter 8
14. **The need for different types of software system and their attributes**: Component 2 Chapter 7
15. **Practical programming**: Component 3
16. **Data security and integrity processes**: Component 2 Chapter 8
17. **Economic, moral, legal, ethical and cultural issues relating to computer science**: Component 1 Chapter 9
If you’re co-teaching AS and A Level in England, we’d suggest that you approach the content in this way for the first year of teaching, and then review it when you teach the added A Level content in the second year.
What your students need to know

- A generalised computer system consists of a CPU (central processing unit), RAM (random access memory), bus system and input and output interfaces. Students should be aware of the basic functions of these key components, i.e. the CPU executes programs, RAM stores currently executing programs, the bus system carries data, instructions and memory addresses between the CPU and RAM, and input and output interfaces allow for communication with peripheral devices such as keyboards, monitors and hard disc drives.

- A von Neumann processor consists of a single Arithmetic Logic Unit (ALU), a single Control Unit (CU) and a single Memory Unit (MU).

- The ALU is where arithmetic and logical operations are performed. The result is stored in a special register called the accumulator (ACC).

- The CU controls the timing of operations by generating a clock signal. Operations take place at regular intervals according to this signal. The CU also decodes the opcode part of instructions.

- The MU is where programs are stored when they are ready to run. Each instruction and item of data in the program will have a memory address associated with it.

- All of the components inside the CPU are connected by the bus system. The address bus carries memory addresses from the memory unit to registers in the CPU. The data bus carries data and the
control bus carries control signals.

- There are different types of CPU. To increase efficiency, an additional co-processor may be installed to carry out specific operations, for example, floating point arithmetic. Parallel processors use multiple CPUs to complete more tasks per unit time. Array processors or vector processors all work together on the same task. Multi-core processors can support multiple paths of execution (multi-threading). An example would be a modern Quad or Dual Core CPU.

- Cache memory is situated on the CPU and, in modern computer systems, also on the motherboard. Recently executed instructions are copied from RAM into cache memory so that they can be accessed quickly if needed again, for example, if a loop is being executed. Random Access Memory stores currently executing programs and is essential. Cache memory improves system performance but is not essential. Read only memory (ROM) stores the programs required when the system boots up. The bootstrap program will test the computer hardware is functioning correctly. The Basic Input Output System (BIOS) stores system configuration information including the boot order of storage devices to attempt to load the operating system from.

- Parallel processors increase efficiency by permitting multiple execution pathways. However, programs need to be written to utilise threads. Most programs cannot be written to utilise threading for the entire run of execution. For example, in a game program, rendering every pixel on the screen can utilise parallel processing but calculating the game score is a straightforward, sequential process. Even with a complex processor architecture, with a high degree of parallelism, a point is reached where the program speed of execution cannot be increased any further. This can be illustrated using Amdahl’s Law.

- The fetch-decode-execute cycle (FDE) is the process by which programs are run. In the first part of the cycle, fetch, an instruction is copied from the memory unit into a special register called the current instruction register (CIR). The assembler needs to work out from the mnemonic which operation it represents. This is the decode part. Finally, some data (operands) will be fetched from the memory unit and copied into the memory data register (MDR). The instruction is then carried out on the operands in the ALU and the result stored in the accumulator.

- In most cases, programs stored in the memory unit run sequentially, starting at the lowest address and using every memory address to the end of the program. A register called the program counter stores the address of the next instruction so the CPU knows where to look. The memory address register stores a copy of the current memory address being worked on. When a program contains a branch or jump instruction, it will also specify the memory address to go to. This will be seen in selection and iteration constructs.

- Each CPU has an instruction set that it supports. CPUs can be programmed in binary (machine code) or a low-level language called assembly language. This uses mnemonics to represent the instructions. The instruction set, and therefore the assembly language, will differ slightly between CPUs. The CPU will need to have a program called an assembler to convert the assembly language code into machine code. Programming in assembly language is time consuming as many lines of code are needed to accomplish simple tasks, for example, adding two numbers together and storing the result. However, programming in machine code (binary) is even more cumbersome. Assembly language is more convenient for programmers but of no benefit to the computer.

- Assembly language uses simple mnemonics to represent machine instructions, for example, ADD, SUB, MUL and DIV. Programs are written to include the register or memory address of the data. Labels are user defined identifiers. Program flow is often sequential but selection and iteration constructs can be brought about using branch (BRA) and jump (JMP) instructions. The Little Man Computer is an assembly language simulator that is a useful tool to aid understanding.

- Input and output devices allow for the transfer of data between the user and the computer. There are a wide range of input devices that aim to make interaction as natural and straightforward as possible. Specific processes and applications benefit from the use of specific types of input, for example, smartphones use capacitive touch screens, examination marking systems use optical mark recognition (OMR). These are just two examples and students should be aware of a wide range of devices.

- Voice input is used to give specific instructions to the computer system and is commonly used in smartphones. Vocabulary dictation allows users to enter data that will be output as a text document. This is of particular benefit in mobile computing and for visually or motor impaired users. Voice print
recognition analyses the pattern of a sound to authenticate a user as part of a security system.

- Secondary storage in a computer system allows data to be stored permanently. Conventional computer systems may contain a magnetic hard disc drive and an optical disc drive, i.e. CD, DVD, Blu-ray. Mobile computing devices are more likely to use solid state storage which is lighter, smaller and has no moving parts. Optical and solid state media are also used for removable media to transfer data between computers.

- The physical organisation of the file system on a magnetic storage device is complex. Files may be modified and additional parts stored in discontiguous clusters on the disc. This is fragmentation and reduces performance as all parts of a file must be located before it can be opened and this increases latency. Defragmentation is the process of moving files and their additional parts (called extents), to occupy contiguous clusters as far as is possible. Defragmentation can decrease the seek time and so, from the user’s perspective, the file will load more quickly. Solid state media does not require defragmentation because reading from random blocks is no slower than sequential reads. Solid state media has a finite number of read/write operations. The defragmentation process would effectively shorten the life of the device so is of no benefit.

- Networks consist of two or more computers, a shared network medium, which may be wired or wireless, and a communication agreement or protocol. A small network, usually within one building and owned by one organisation, is called a LAN (local area network). A larger network, which can span a whole country or even the entire globe, and may be owned by many organisations, is called a WAN (wide area network).

- A client is a computer that accesses network services. A server is a computer that provides network services, for example, file server, mail server, web server, etc.

- In a peer-to-peer network, each computer can provide or receive network services.

- In order for communication to occur between devices, there needs to be an agreement on the network medium being used, how network nodes are addressed, the format and structure of data being transmitted, speed of transmission, error correction and any security measures (for example, encryption) used. It is important that both the sender and recipient agree on the protocol to be used. Many network protocols exist but the best known belong to the TCP/IP protocol suite.

- Below is a table summary of the function of protocols that your students need to know:

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP (Hypertext Transfer Protocol)</td>
<td>Transfers web pages from a web server to a client computer’s web browser.</td>
</tr>
<tr>
<td>FTP (File Transfer Protocol)</td>
<td>Transfers files between server and client.</td>
</tr>
<tr>
<td>SMTP (Simple Mail Transport Protocol)</td>
<td>A protocol for sending e-mails from a client to a server and for relaying between servers.</td>
</tr>
<tr>
<td>IP (Internet Protocol)</td>
<td>Delivers data packets.</td>
</tr>
<tr>
<td>TCP (Transmission Control Protocol)</td>
<td>Adds reliability to packet delivery.</td>
</tr>
<tr>
<td>IMAP (Internet Message Access Protocol)</td>
<td>Allows e-mail messages to be accessed online.</td>
</tr>
<tr>
<td>DHCP (Dynamic Host Configuration Protocol)</td>
<td>Dynamically allocates IP addresses to host computers.</td>
</tr>
<tr>
<td>UDP (User Datagram Protocol)</td>
<td>An alternative transport layer protocol to TCP that delivers packets quickly but unreliably as it does not use acknowledgements and is connectionless (does not set up a pathway prior to data transmission). The application layer protocols that depend upon it are those that often send out repeated messages, for example, DHCP requests and routing updates.</td>
</tr>
</tbody>
</table>

- Handshaking is the process that takes place as a precursor to data transmission. During a handshake, the speed of transmission, error correction, and character set to be used are agreed.

- Wireless networks are very commonplace. Mobile computing devices such as a smartphone or tablet computer, can connect to a wireless LAN. Many modern applications require a network connection, for example, messenger apps. Apps can also be used by network technicians to monitor associated clients, signal quality and strength and to locate wireless “dead spots”.

© Cambridge University Press
Wireless access points (WAPs) are network devices that generate a wireless signal. Mobile computing devices will have an integrated wireless network adapter and will automatically scan for wireless networks. The WiFi protocol is covered by the IEEE 802.11 standard. Public WiFi “Hot Spots” have open networks which require no authentication before use (although many require a subscription service to access the internet). Private wireless LANs are protected by a network key and encryption. WEP (wired equivalent privacy) uses a static key to encrypt data. As the key value does not change, WEP is vulnerable to cracking (illegal discovery of the network key and decryption of data). WPA2 (WiFi protected access) is a great improvement as, although the key is still used, a protocol is used to change it during communication making it exceedingly difficult to crack. Additionally, WPA2 supports much stronger encryption called AES (advanced encryption standard) that further secures communication. Large organisations may use a whole infrastructure security protocol called 802.11X which allows users to be authenticated onto the network system against a login server much the same as they would if they logged into a wired computer.

Joining a wireless network occurs in distinct stages. Firstly, the wireless host will scan for available networks. If one is found, it will associate with the network and will then attempt to authenticate either by the user entering in their username and password or network key. For convenience, these details may be stored on the mobile device and used to authenticate automatically after association. If a mobile device loses the signal from a Wireless Access Point, it becomes disassociated from the network.

Common misconceptions

- Confusion between the purposes of specific registers within the CPU can occur. Similarly, between processor architectures. It would be helpful for students to produce a glossary, or their own Wiki, to work collaboratively on definitions of key terms.
- Selection and iteration constructs can be tricky for students learning assembly language programming using the Little Man Computer simulator. Provide plenty of opportunity for students to experiment by creating programming challenges.
- Provide clear definitions for voice input, voice recognition and vocabulary dictation systems to make distinctions between these three technologies distinct.
- The topic of communication has a number of acronyms associated with it. It is helpful for students to produce a glossary to aid remembering. They also need to understand the process of handshaking and why it is necessary before data can be exchanged.

Lesson activity suggestions

Physical CPUs for students to look at is a useful activity. If possible, try to find examples of different generations and from different types of devices. An e-mail request to colleagues may produce some interesting specimens and none of the devices need to be functional. It is useful to include some examples of System on Chip devices (SoC) which combine CPU, RAM and GPU into a single chip. These are commonly found in small systems such as a hand-held games console or mobile phone. These usually prompt some interesting questions such as, “Where is the RAM?”

Students may find the micro-architecture of the von Neumann CPU difficult to grasp as there are a number of new acronyms and they need to appreciate how each component interacts with the others. An interactive and fun way to do this is to build a ‘jigsaw’ model of a von Neumann CPU. This encourages them to think about connections within the CPU and the fetch-decode-execute cycle.

If your students have access to Raspberry Pi computers, these can be safely configured for overclocking and performance improvements can be measured.

The range of processor types can be investigated from looking at the range of tasks that they are expected to perform. For example, students will appreciate that an array processor will be more efficient at processing graphics in a game than a single core von Neumann CPU.

The Little Man Computer is a CPU simulator that can be downloaded or accessed online as a Java applet. This is a very concrete way for students to understand the link between a program written in low-level code and operations in the CPU to produce the output. When students are familiar with the instruction set of the Little Man CPU, they can begin to appreciate the components required to build a working program.
The card game, ‘Whose Line of Code is it Anyway?’ is a fun way to reinforce these concepts. Produce a deck of cards with a single LMC instruction on each one. Organise students into groups of two, three or four students. Deal out the cards. Each student must lay an LMC instruction when it is their turn that will ultimately produce a program that will compile. When a student is certain that the program is complete, and will compile, another student will 'test' the program using the LMC simulator. If it does compile and can be executed, the student declaring completeness wins the round. All of the cards are collected up again and shuffled. The game ends when a student has won three rounds.

Students can assume the role of a CPU component and work collaboratively to ‘execute’ a simple program. This activity reinforces the sequential nature of processes. This also assists with understanding the name of specific registers and their role.

**Topic: Processor components**

**Starter:** CPU ‘show and tell’.
Examples of processors in different devices, for example, desktop PC, laptop, smart phone, games console, Raspberry Pi.

**Main activity:**
Paper ‘jigsaw’ of components of a typical von Neumann processor. Identify the purpose of each component. Completed jigsaws could be wall-mounted as a poster.

**Plenary:**
Discuss how processors have evolved to increase system performance, for example, multi-core, multi-threading and increased levels of cache memory.

**Raspberry Pi activity:**
Investigate overclocking using the preset values available in the configuration panel.

**Follow-up ideas/Homework**
Research project on use of CPUs and microprocessors in mobile phones and games consoles.
Research the contribution of Jon von Neumann.

**Topic: Processor architecture**

**Starter:**
Processor architecture and application ‘walking wall’ including von Neumann, vector, parallel.

**Main activity:**
Little Man Computer investigation. Basic programming including challenges card game involving LMC instructions of iteration and selection. Students must lay cards in sequence to make a correct program.

**Plenary:**
Discuss the difference between a CPU, microprocessor (for example, in a washing machine), and System on Chip (SoC), for example, in a Raspberry Pi.

**Follow-up ideas/Homework**
Practise examination questions.
Little Man Computer programming challenge: multiplication and division using the LMC.

**Topic: The fetch-decode-execute cycle**

**Starter:**
What is an instruction? How do we carry out a sequence of instructions? Illustration using a task to make an origami paper model.

**Main activity:**
Whole class activity – simulating a CPU. Each student assumes the role of a component and runs a simple program to add two integers.

**Plenary:**
Discussion: is it better to have many simple instructions (RISC) or fewer, more complex instructions (CISC)?

**Follow-up ideas/Homework**
Investigating interactive resources to learn the fetch-decode-execute cycle.
Produce a comparison of RISC and CISC architectures and examples of computers that use each.

**Topic: Networking**

**Starter:**
What is a protocol? Students work in teams to develop a communication method that must be secure. Students should be encouraged to think about what they will do if the transmitter is sending data too fast? What will they do if an error occurs? They must ensure that a ‘hacker’ will not be able to work out what the message is by eavesdropping on the conversation.

**Main activity:**
Identify protocols that relate to physical standards, for example, Ethernet, and those that are entirely implemented in hardware.

Investigate a number of key protocols: IP, TCP, RIP, ICMP (PING), DNS, DHCP, HTTP, FTP, SMTP, IMAP. Each student could produce a wiki page about their protocol. (If you use the VLE Moodle, wiki is a built-in tool.) Alternatively, Google Slides could be used or other presentation software.

**Practical activity:**
If resources are available, create a small LAN using two clients and use Wireshark to capture live protocol packets in real time. This activity can also be carried out on a Raspberry Pi although use of Wireshark is not recommended. Use the terminal based Tshark instead or tcpdump and export to a file which can be read using Wireshark on a laptop. Wireshark is a powerful protocol analyser. Passwords sent in clear text will be visible. It is important that it is only run on a lab setup and not on the school’s network.

**Plenary:**
Socrative or similar online quiz with questions relating to protocols.

**Follow-up ideas/Homework**
Investigation into the changes between IPv4 and IPv6.

Produce a mind map of protocol concepts. This can be carried out using online software such as Popplet.

**End-of-chapter answers**

1

(a) A hub will allow multiple networked devices to be connected together. Packets sent to the hub will be broadcast to every other connected device.

(b) A switch will allow multiple networked devices to be connected together. Packets sent to the switch will be routed to the correct port using their MAC addresses.

(c) A router behaves in a similar way to a switch. However, routing is done using IP addresses and allows a LAN to be connected to a WAN, like the internet.

2

If there is a low bit rate then smaller amounts of data can be transmitted per second. When streaming video, if the bit rate falls below the amount needed to show a single second of footage then the video will stutter or start buffering. When downloading files, the bit rate directly impacts the length of the download for large files. For small files, the bit rate is not as important.

3

On the top layer is the application layer which is where applications will format their packets before it is sent. Web browsers will use HTTP (hypertext transfer protocol) while file transfer uses FTP (file transfer protocol). In order for the packet to be understood, both sender and receiver must be using the same protocol. When the packet has been correctly formatted for the destination software, it must then either use the TCP or UDP protocol. TCP allows error detection and packets to be ordered correctly. UDP, on the other hand, does not have any verification checks. Finally the packets have routing information added by the IP protocol before being placed onto the network hardware.

4

Data is fetched into the CPU by first copying the PC into the MAR. The address from the MAR is used to fetch data into the MDR while the PC is incremented to point to the next instruction. The instruction is then copied
from the MDR to the CIR (current instruction register) ready for execution. As the instruction is being decoded and executed the next instruction is fetched. Pipelining is where the different stages of the FDE cycle are done in parallel, only on different instructions. As one instruction is fetched, another could be decoded and executed. This helps reduce the delay caused by fetching. However, if a jump command occurs then fetching the net instruction will be a wasted operation as the value of the PC will change. This is why any form of branching is considered to be a computationally expensive operation.
Chapter 2: Data transmission

LEARNING OBJECTIVES

• 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.
• 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.
• Calculate data transfer rates on a network.
• Calculate lowest cost routes on a network.
• Describe the internet in terms of a world-wide communications infrastructure.

What your students need to know

• Serial data transmission sends data one bit at a time in sequence down a single wire. An advantage is that only one wire is required. This can reduce costs and the physical space required. It may appear that serial communication could be slow but the clock used to control how fast data can be sent can be increased to send more data per unit time. Parallel data transmission uses multiple wires or channels to send many bits at once. Early parallel communication systems were faster than serial links. However, this is not now the case. Parallel communication can also introduce interference due to induced currents between wires called crosstalk which reduces signal quality.

• Simplex data transmission takes place in one direction only. An example of this is a conventional television broadcast. Half duplex data transmission is two way communication but only in one direction at a time. A suitable example would be a ‘push to talk’ system, for example, a walkie talkie. Full duplex is two way communication but can occur in both directions simultaneously. An example would be a real time communication system such as Skype or Facetime.

• Network links support a certain range of communication frequencies or channels. This is the bandwidth and represents the total theoretical data carrying capacity of the medium. In reality, there are always reasons why the actual data rate, or throughput, is less than this. One reason, network collisions, will be explained in greater detail in the next section. To increase efficiency, multiple signals can be combined into one and sent together. This is called multiplexing. Commonly used methods are frequency division multiplexing (FDM) and time division multiplexing (TDM).

• Wide area networks (WANs) were originally based around the telephone system. Packets of data were sent along connections that had been previously set up. All packets would travel along the same path. This is called circuit switching. If a pathway became congested, transmission would be very slow. Some pathways that could relieve the pressure may have been underused. Modern systems use devices called routers to carry out packet switching, which is the process of sending each packet along the next step path by making decisions on the most efficient route.

• The TCP/IP protocol suite is by far the most common and is used on both LANs and WANs (including the internet). Data is transmitted in units called packets. Each packet will have the same structure and contain specific fields. Below is an example of a stylised IP packet:

<table>
<thead>
<tr>
<th>IP packet header</th>
<th>Source IP address</th>
<th>Destination IP address</th>
<th>TCP header</th>
<th>Sequence number</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Ethernet LAN networks allowed any node to transmit data; all nodes could see the packet but only the destination node would actually process the message. This is called broadcasting. When two nodes send data at the same time, the signals will interfere with one another. This is called a collision. A protocol called carrier sense multiple access/collision detection (CSMA/CD) determines how collisions are detected and handled. Nodes ‘listen’ to the network and can only transmit when it is quiet. However, if two nodes begin to transmit at exactly the same moment, a collision will occur and a
jamming signal is sent to prevent further transmissions. Both nodes will wait a random amount of time and then attempt retransmission. As the number of collisions increases, network performance of the network is degraded. As the number of nodes on a LAN increases, so do the number of collisions.

- **Routing** is the process of path determination for a packet. Routers hold data in a routing table of all of the networks that they are connected to or have learned about from updates from other routers. Routing protocols calculate the optimum path based on various metrics (measures). The most simple is RIP (routing information protocol) which uses **hop count** as the metric (how many router ‘hops’ are between the packet and its destination network. The lowest metric value represents the lowest cost path. Other more sophisticated protocols such as OSPF (open shortest path first) use a complex metric to calculate path cost including link bandwidth, reliability and round trip time.

- Data transfer rates are expressed in Mbps (megabits per second). If a link has a bandwidth of 100 Mbps and we wish to send 1 gigabyte of data, students should be able to convert units to deduce the correct time. For example:
  - 1 GB = 1024 x 1024 x 1024 bytes = 1 073 741 824 bytes
  - There are 8 × bits in each byte so this represents:
  - 8 × 1 073 741 824 = 8 589 934 592 bits = 8 589 934 592/(1000 × 1000) MB = 8590MB
  - The link speed is 100 Mbps (assuming no latency):
  - Time in seconds = 8590/100 = 85.92 seconds (1 minute 26 seconds)
  - To aid understanding, provide students with practice questions involving links of different speeds and different units of data (KB, MB, TB, etc.)

- Path cost can also be calculated. In the graphic below, path costs are shown near the link.

If a data packet is to be sent from A to E, the table below shows the total cost for each possible path:

<table>
<thead>
<tr>
<th>Route</th>
<th>Path cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A to E via B and D</td>
<td>6 + 7 + 4 = 17</td>
</tr>
<tr>
<td>A to E via C and D</td>
<td>3 + 4 + 4 = 11</td>
</tr>
<tr>
<td>A to E via B, C and D</td>
<td>6 + 2 + 4 + 4 = 16</td>
</tr>
</tbody>
</table>

From the above table, it can be seen that the second cost is lower and therefore represents the desirable route.

- The internet is a global, public WAN. It is not owned by one organisation but is maintained by many. Routers are concentrated into points of presence all around the internet. These are the sites of packet switching. Much of the infrastructure is based on fibre optic cable which transmits data at the speed of light. The World Wide Web is the network of web servers on the internet. Organisations are identified by their **domain** name which is a string that represents their range of external server IP addresses. Domain names have a specific structure with the top level domain on the rightmost side of the address, for example, [www.wjec.co.uk](http://www.wjec.co.uk). WWW denotes that the server is a web server. The whole domain name is wjec.co.uk in this example. The protocol DNS (Domain Name System) translates the URL (Uniform Resource Location) entered by a user into the address bar of their browser, by performing a lookup operation in its database. If it is unable to resolve the URL, it will query another
DNS server. When a match is found, the IP address provides a simple, hierarchical naming system for devices and resources.

**Common misconceptions**

- Students need to have a good understanding of the operation of CSMA/CD and its role in handling data collisions. It is worthwhile investigating the evolution of Ethernet networks and issues caused by increased numbers of clients and the subsequent increase in collisions. It is often helpful to provide some real world examples such as a noisy classroom. When multiple people are speaking, the message cannot be heard. What is needed is somebody to tell everybody to stop speaking (the jam signal) so the ‘network’ becomes quiet and then speakers wait until it is quiet for them to begin speaking again.

- Calculations relating to data transfer rates can lead to errors due to converting between units, for example, megabytes to bits, etc. It is helpful to remind students of the relationship between units of data storage and that calculations should be worked through sequentially by converting unit of storage to bytes first, then to bits, then using the data transfer rate to calculate the time taken. Calculations relating to path costs are more straightforward.

**Lesson activity suggestions**

**Topic: How data is transmitted**

**Starter:**
On mini whiteboards or paper, students should consider the steps involved in sending an e-mail. They may not know the precise process or names of protocols at this stage but they may be able to pick up on certain features such as delivery receipts and delay notifications. Use Q and A to elicit as much detail as students know at the start of the lesson.

**Main activity:**
Students will investigate LANs and WANs and the role of clients and servers. They should consider how digital signals can be modulated to carry data on copper media as electrical signals, fibre optic cable as light pulses and as radio frequency waves in wireless networks. They should also explain multiplexing, circuit and packet switching. They can use this information to produce a fact sheet about networks to be given to householders by a telephone company/ISP.

**Plenary:**
Discussion about the measurement of the speed of a network. They will probably have encountered the term megabits per second (Mbps) in advertisements and in relation to mobile internet access but this should be explained to them as part of their studies. Consider how fast network signals can travel. What are the limitations with data transfer rates? This will permit the definition of two important terms: bandwidth and throughput.

**Follow-up ideas/Homework**

Produce an illustrated poster or animation to show the meaning of the terms simplex, half duplex and duplex, serial and parallel communication.

**Topic: Active learning session**

**Starter:**
How is data transferred? Students consider the following network concepts and protocols: CSMA/CD, packet switching, multiplexing, routing and path determination.

**Main activity:**
Students work in small groups to develop a simulation of a network concept listed above. They then present their simulation to the rest of the group.

**Practical activity:**
If resources are available, set up a Raspberry Pi network or use a simulator to set up and view routing, address resolution and network statistics. The command `ifconfig eth0` will show current configuration of an Ethernet port. `Route` or `netstat -r` will show routing table entries. `Traceroute ip address` shows hops to a destination address. The `nslookup URL` command will show IP addresses mapped to the URL.

**Plenary:**
Provide students with network diagrams showing link costs. Students decide on the best route to the destination and calculate the metric. Static routes (which always have a metric of 1) can also be added in so students can see how these will always be used in preference.

**Follow-up ideas/Homework**

Investigate *Dijkstra’s algorithm* as utilised in the OSPF routing protocol. What are the flaws of RIP and how does OSPF improve upon these?

**Topic: Network models**

**Starter:**
Recap physical and logical protocols.

**Main activity:**
Run through the two key networking models (TCP/IP and OSI) with reference to their age and purpose.

Students complete a mix and match activity of layers and protocols at each layer. For this activity, use the TCP/IP model as it is simpler.

**Plenary:**
Discuss the importance of standard interfaces. Changes in physical protocols do not affect logical ones and vice versa.

**Follow-up ideas/Homework**

Produce an animated presentation or poster to illustrate example communications between a client and a server. Students could select one application layer protocol, for example, HTTP, DNS, DHCP, etc.

**Topic: Data transfer**

**Starter:**
Recap units of data: bit, byte, kilobyte, megabyte, gigabyte, terabyte. Students produce a table to compare measurements.

**Main activity:**
Students to complete activities to calculate times taken to transfer data across a network link, for example, a 1 MB file across a 100 Mbps link. When students have completed some structured calculations, a group Kaboodle quiz can add a fun, competitive element. Alternatively, an animated, interactive quiz, for example, using presentation software or a Google form.

**Plenary:**
Discussion to consider managing a fictional WiFi network and providing sufficient bandwidth to: stream video, make online calls, view web pages and send e-mail. By the end of the session, students should understand that bandwidth is a finite resource and that some activities require higher data transfer rates than others. They should also appreciate that the more people that join a wireless network, the less bandwidth is available to each of them.

**Follow-up ideas/Homework**

Investigate how signal modulation affects data transfer rates. Research how emerging standards are providing more bandwidth. A less technical task could be to consider trends in mobile computing and how wireless networks can match speeds provided by a cabled connection.

**Topic: Circuit and packet switching**

**Starter:**
Present students with a maze game in which they must move four counters from the ‘source’ to the ‘destination’ in the smallest number of moves using any route. They must then repeat the activity but can only move the counters along the same pathway once they have decided which they think is the shortest. This will help them to consider the advantages of packet switching over circuit switching.

**Main activity:**
List different types of data that would be sent across a network which must be streamed (video and audio) and which can be split up into packets that can be sent separately (e-mail, webpage, file transfer).

Explain the historical basis of internet traffic using the telephone network and then transition to packet switched technologies.
Divide the class in two: one to represent circuit switched networking, the other packet switching. Each half must prepare a 'pitch' to promote their technology.

**Plenary:**
Complete example examination questions and peer mark.

**Follow-up ideas/Homework**
Students to research VoIP and the convergence of data and voice networks.

**End-of-chapter answers**

1
(a) Packets can be sent along a cable in one direction only at any time.
(b) Duplex allows packets to travel in both directions at the same time.
(c) Bits sent in parallel are sent at the same time, rather than one after another. The duplex mode can be set on top of this.

2
Each row and column has a parity bit added to help locate when a bit is in error. Row 3 has an odd number of bits set to 1, which means an error happened on this row. The fourth column also has an odd number of bits which means that the bit in row 3 column 4 has an error. Because we know where that bit is we can swap it from 0 to 1 in order to correct the error.

3
In packet switching, the packets will arrive in a different order while circuit-switched networks' packets will always arrive in order.
In a circuit-switched network if part of the line breaks then communication will fail while packet-switched networks will find alternative routes.

4
3/8 = 0.375 Mb/s
5/0.375 = 13.3 seconds
**Chapter 3: Data representation and data types**

**LEARNING OBJECTIVES**

- Explain the terms bit, byte and word.
- Describe and use the binary number system and the hexadecimal notation as shorthand for binary number patterns.
- Describe how characters and numbers are stored in binary form.
- Describe standardised character sets.
- Describe the following different primitive data types: Boolean, character, string, integer and real.
- Describe the storage requirements for each data type.
- 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 explain their effect upon accuracy.
- Explain and use shift functions: logical and arithmetic shifts. Interpret and apply shifts in algorithms and programs.
- Describe the causes of overflow and underflow.

**What your students need to know**

- Electrical current can either flow through an electronic circuit or be stopped, which a computer represents with the digits 1 for ‘ON’ and 0 for ‘OFF’ that form the binary number system.
- Each 0 or 1 is known as a bit (binary digit) and it takes large groups of these digits to represent any significant amount of data. 4-bits are called a nibble, and 8-bits are called a byte. Other units are kilobytes, megabytes, gigabytes and terabytes.
- A word is not a fixed unit but depends on the CPU and how many bits it can manipulate at one time.
- Internally all characters are represented as binary. The two most common character sets are ASCII and UNICODE, where ASCII is a 7-bit code with smaller set of characters, only really good enough for English characters. Extended 8-bit ASCII added a few more characters that were still not good enough for the world’s languages other than English and also were implemented differently in different systems. UNICODE uses up to 32 bits to represent many times more characters, including all living and dead languages’ writing.
- Students need to know the base numbers representing various degrees of 2: 1, 2, 4, 8, 16, etc. Any decimal number is converted to a binary by subtracting these base numbers until we are left with 0. Converting binary to decimal involves adding these numbers together (only if there is a binary 1 corresponding to their position in a byte).
- They also need to know that the number of bits allocated to storing a particular decimal number will influence the maximum size of the number that can be stored in that memory.
- There are few ways to represent negative numbers in binary: some of the more common are ‘sign and magnitude’ and ‘two’s complement’.
- Sign and magnitude is the easier one of the two; the leftmost digit (otherwise known as MSB, most significant digit) becomes an indicator for a sign: 1 for negative and 0 for positive. By dedicating a whole bit to a sign, we halve our range of numbers that can be stored in the same amount of memory, so two’s complement is a more economical solution, which, however, requires more processing.
- To convert a negative decimal number to a two’s complement binary one, we first write it out as if
it was positive. Then, starting on the right, leave every bit up to and including the first 1 alone, but flip/invert others from 0 to 1 and from 1 to 0. This is known as complementing, as 0 and 1 together complete the binary set, so they complete or ‘complement’ each other.

• Binary addition isn’t that different from decimal addition of long numbers. One has to remember to carry, which is quite common.

• Binary subtraction is the same as binary addition but the number to be subtracted is converted into a negative number using two’s complement.

• Hexadecimal numbers have base 16, rather than base 2 of binary numbers. Therefore, each hex digit is equivalent to a nibble. Two hex digits make up an equivalent of a byte. To convert a decimal to a hex number, you might need to convert this number to an 8-bit binary and then split that into two nibbles, 4 bits each, and convert them individually to hex.

• The floating point is a binary version of the scientific notation you can get in your calculator, made up of the mantissa and the exponent. The number of digits is fixed but the decimal point can float around. In binary, both the mantissa and the exponent are in binary, obviously. The mantissa is ‘normalised’, that is, always written with 01 as the two leftmost digits with a floating point in between the 0 and the 1. The exponent, once converted to decimal, tells us how many times to shift the floating point left or right (depending on whether the exponent is positive or negative) to get the actual number we have originally stored.

• The number of bits that represent the mantissa determine its precision – how small a fraction can we store without truncating the really small ones. The number of bits in the exponent determines range – the maximum and the minimum numbers we can represent in a given amount of memory.

• Fractional numbers, both positive and negative, can be represented.

• Not all fractions can be represented in binary; some require rounding.

• When adding two floating point numbers, their exponent must be the same so we must move the floating point in the mantissa accordingly by adding 0s from the left if needed. For subtraction, we change the mantissa of the number that’s being subtracted to the negative.

• Students need to be able to compare floating and fixed point systems and list advantages and disadvantages of each.

• Computer storage is limited and the compromise is that very small and very large numbers may not be stored correctly, due to the insufficient number of bits, resulting in the underflow and overflow, respectively.

• Bitwise manipulation and masks is applying logical operators AND, OR, NOT and XOR to every digit in a binary number. To apply these to a number, we need another number, called a mask, so that each digit of the mask number will interact with a corresponding bit of the original number depending on the logical operator applied, for example, 1 in the original number AND 1 in the masking number will produce 1, 1 and 0 will produce 0. Students need to know the effect of all of the logical operators.

Common misconceptions

• Hex is not used by computers, but by humans working with them to reduce errors reading long binary numbers.

• Students often forget about their carry bits when doing binary addition and subtraction.

• They can also forget which bits to flip and which ones not to flip when applying two’s complement.

• Students can mix up mantissa and exponent, as well as forgetting to normalise the mantissa.

• They can forget what logical operators do, especially confusing OR and XOR, as well as forgetting that the NOT operator only needs one number, not two.

• Students need to remember that ASCII only handles English letters and they can often forget how many bits are in each character set.

• Truncation is not rounding; truncation is always rounding down.

Lesson activity suggestions

Topic: Binary

Starter:
Reminder why computers use binary. Using a Raspberry Pi computer’s GPIO pin and a voltmeter, use a Scratch GPIO or Python program to supply power to the GPIO pins and then turn them off. Have a volunteer student with an electric multimeter verify how much voltage is supplied to a pin when it’s ‘on’.

**Plenary:**
Have a ‘dictation’ where a decimal number is read out, students write down binary (or hex) versions; a few seconds later, the teacher moves to the next number. Students’ papers are then swapped and peer marked.

**Follow-up ideas/Homework**
Create a next dictation for the teacher to read out in the next class.

Let’s suppose that, instead of 0s and 1s, we used different numbers, for example, 1 and 5. Recast decimal numbers to binary to be expressed using 1s and 5s. For example, a simple number like 12 would be expressed as 1100 in regular binary but as 5511 in our imaginary binary. This is to demonstrate that the symbols themselves are not important, they are just placeholders. Alternatively, we could use any characters instead of 0s and 1s for this activity.

**Topic: Fixed vs floating point storage**

**Starter:**
Revision of ‘scientific’ notation in Maths.

**Main activity:**
Doing multiple examples of decimal or hex to binary (floating and/or fixed point) conversion.

**Plenary:**
Using Audacity, teacher saves a 16-bit (CD quality) file in 24-bit – students are asked if there is any improvement. Then it is saved to 8-bit, students are to comment on the difference in sound.

**Follow-up ideas/Homework**
Software that allows music recording is known as a ‘Digital Audio Workstation’. Some of them store digital music in floating point (Cubase) and others (Pro Tools) store them as fixed point. Investigate the advantages and disadvantages, using resources like this one: Steven W. Smith, *The Scientist and Engineer’s Guide to Digital Signal Processing, Chapter 28: Digital Signal Processors*

Investigate how digital audio is stored in the PCM format.

**Topic: Bitwise manipulation and mks**

**Starter:**
Demonstration of two layers interacting via a mathematical operation where one layer masks the other.

**Main activity:**
Students use a programming language like Python that supports binary to decimal and back conversion and also shows floating point binary numbers. Python also can implement all logical operators.

**Plenary:**
Look at the concept of ‘subnet masking’ which allows routers to have more devices on the internet than available IP addresses using TCP/IP.

**Follow-up ideas/Homework**
Create a program that creates sample questions on binary addition/subtraction, and other binary operations and supplies correct answers – ideally, writing all this in to a text file.

Research masking [here](#). For bitwise numbers and masking, a good exercise could be to use either Photoshop or its free alternative GIMP to create two layers of graphics and then cycle the blending mode between the layers which represent variable masking operations. There is also a masking feature which applies brightness or other adjustments to only masked portions of the image, as shown on the pictures, using Photoshop CS2.

What you see here is the layer blending palette. Layers are used in graphics software to separate elements, for example, background and foreground and apply different effects to them. Good graphics feature multiple layers blended in. When blending multiple layers into one image, it is sometimes important to only keep the darker pixels or the one of a certain colour. This palette uses bitwise operations on pixels (which are internally just binary numbers representing colour and transparency of a pixel). Can you determine which
bitwise operation is performed by each of these modes, for example, ‘color dodge’, etc? (One of the possible answers: ‘Darken’ subtracts the binary brightness value of the blended layer from that of the layer below it.)

In addition to blending of layers, Photoshop (and many other packages) support masking. This is when instead of erasing a part of an image we cover it up with a different layer or ‘mask’. This way, we can always go back to the original image without trying to restore what we erased. On the images below, we created a Layer mask where white lets the layer’s image through, black cuts a hole in it and grey covers semi-transparent blending. A good example of this technique can be seen [here](http://www.photoshop.com/tutorials/layers-mask-tutorial): PhotoShop Layer Mask Tutorial by Devvyn Murphy.

Which bitwise operation would that be? (Answer: binary addition. It is used for various masking operations in general.)
**Topic: Character sets**

**Starter:**
Open a foreign language site that uses non-Latin characters, for example, Bulgarian or Chinese, then force change the character encoding. Bulgarian Cyrillic characters immediately can become Chinese, but it’s not Chinese text that makes any sense!

**Main activity:**
Presentation of memory requirements of ASCII vs Unicode UTF-8.

Look at this [website](see link in plenary) and various sets available. Consider the use of the Windows’ bundled program ‘Character Map’ (or Insert Symbol in MS Word) which also demonstrates various subsets of characters.

**Plenary:**
Treasure hunt: find Unicode characters for some obscure symbols, for example, Egyptian hieroglyphics or a particular Chinese character. Reference for Unicode is [here](here).

**Follow-up ideas/Homework**
Research: it is widely said that Unicode was created to serve the WWW. Would Unicode have been invented without the arrival of the Web?

**Topic: Overflow and underflow**

**Starter:**
Students are asked to take out their calculators and calculate the sum of 0.00000000001 and 0.0000000000005. Depending on the make of the calculator, some will show something meaningful, for example, in engineering notation, others might show error or zero. Discussion of why calculators have mantissa/exponent notation where a fraction can be shown as ‘5 x 10⁻⁹’.

**Main activity:**
Students practise adding (and subtracting) very large and very small binary numbers). Teacher could provide two binary numbers and without solving ask for a show of hands: how many students believe this will underflow or overflow.

**Plenary:**
A look at the Year 2000 Bug and its causes. (Answer: trying to conserve memory, resulting in the overflow after ‘year 99’.)

**Follow-up ideas/Homework**
Write a program that accepts the input of two binary numbers and if their sum overflows or underflows, the program will display the result with a relevant warning.

**End-of-chapter answers**

1
(a) Left shift by 6.

(b)

**Data**

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

**2**

(a) -12

(b) Largest: 0111 1111 = 127

Smallest: 1111 1111 = -127

(c) 1111 1111 + 0111 1111 should produce 0 (as 127 + -127 = 0)

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The answer is clearly not 0, even after ignoring the overflow. Addition using negative numbers does not work as expected.

**3**

(a) 244

(b) 218

(c) 136

**4**

(a) 119

(b) 3E

(c) 2C

**5**

A floating point number has a 4-bit mantissa and a 4-bit exponent.

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

Mantissa | Exponent

(a) Exponent = 3

Converting mantissa to positive value = 0.111

Moving decimal place = 111.0

Final answer = -7

(b) The range of possible values would be reduced, but the accuracy of the number would be increased (due to larger mantissa).

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

Mantissa | Exponent

(c) It is not normalised.

1.01 should become 0.100 and 2 added to the exponent. The normalised version would be 0100 0011
Chapter 4: Organisation and structure of data

LEARNING OBJECTIVES

- Explain the purpose of files in data processing.
- Define a file in terms of records and fields.
- Describe how files may be created, organised, updated and processed by programs.
- 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.
- 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 purpose of, and be able to use, a hashing algorithm.
- Compare different hashing algorithms.
- Explain the use of multi-level indexes.
- Explain the techniques used to manage overflow and the need for file re-organisation.
- Explain the need for file security, including file backup, generations of files and transaction logs.
- Describe the need for archiving files.

What your students need to know

- Most data is stored by computers in related sets, called ‘data structures’. They come in different types, each suitable for different applications, depending on the data type of the ‘primitive’ elements that make them up (like cells make up living organisms).
- Data on your computer is stored in files which have a structure. While not all of them will follow this structure, we can imagine a grid with rows which are called ‘records’ and columns which are called ‘fields’. Records store all saved information about an object and fields store the data that this information is made of, for example, ID, name, quantity, etc. and can be of different data types, unlike arrays which usually need the same data types for all bits (this can be avoided by storing all fields as strings and then converting when needed).
- When a computer stores records, the operating system needs to know how much memory to allocate for the file, so we need to set up the width of the fields, where units are bytes. Different data types take up different amounts of memory, for example, 1 stored as an integer will take up less space than ‘1’ stored as a string (because most modern computers don’t just use 1-byte ASCII but 2-byte or longer Unicode characters). 123456 will take up less memory if stored as a 32-bit integer instead of a 6+ byte string. Signed and unsigned integers, real numbers with fractions, they take up different amounts of space, too.
- Students need to know how to estimate a file size based on the information recorded and the width of fields and the number of records, allowing for overhead, too.
- Students need to know pseudocode for accessing files: reading, writing, appending, including different types of files: serial, sequential and indexed sequential, as well as random files.
- Serial is the simplest in structure and programming technique. It is not the fastest for large numbers of records. Similarly, students need to know advantages and disadvantages of various types of files access, for example, random is the fastest but wastes space and could end up being slower if the file becomes overfilled and fragmented.
- Hashing functions and their pseudocode are to be learned, as well as the block-based structure of random files.
- A particular technique for merging master and transaction files, explained in the student material,
needs to be practised, especially in pseudocode. This technique combines the advantages of easy and fast recording of serial files with the faster reading of indexed sequential files.

- Students need to know the difference between backing up (making a copy of a file in a safe place) and archiving (moving the file to a safe place) and need to produce pseudocode for backing up and restoring data.

**Common misconceptions**
- When calculating file sizes, forgetting to add overhead or losing track of the data types of fields.
- Mixing up different types of file access, for example, serial and sequential.
- Not seeing the connection between the hashing functions of random files and the concept of hashing from Unit 1.

**Lesson activity suggestions**

**Topic: File design**

**Starter:**
Using the operating system’s built-in property or free software, like ‘Folder Size’ (for Windows), to get how much space different folders take up.

Investigate which file types tend to be larger.

**Main activity:**
Students are shown a spreadsheet with sample data and they have to:
- Determine the size of the file that could result
- Write pseudocode to write and read it to/from a file.
- Alternatively: students are given pseudocode that writes/reads from a file and they need to reconstruct the table structure from the pseudocode given.

**Plenary:**
Teacher reads off a list of applications and students are to supply to the most suitable file format, for example, ‘high scores database’, ‘library catalogue’, ‘saved RPG game’, etc.

**Follow-up ideas/Homework**
- Produce a pie chart of the contents sizes of the students’ memory sticks/network folder.
- Save the same spreadsheet file in various formats allowed by the program and compare their sizes: comment on the overhead.
Another idea: use DB Browser for SQLite this website to convert a CSV file to a SQLite database and compare the size. Open the SQLite file in a plain text editor and try to see where the extra overhead is coming from. It might also be worth exploring some of the things that are created automatically when conversion takes place, for example, a list of tables, an index, etc.

**Topic: File organisation**

**Starter:**
Discussion: Why do books have ‘tables of contents’ or ‘indices’? Could the following books still be useful without one: a novel, a textbook, a dictionary?

When can you have a table of contents of another table of contents (this is known as multi-level indexing)?

**Main activity:**
Discussion of differences and similarities between serial and sequential file access.

Students write pseudocode and then code a program that can update a record in a (a) serial file; (b) sequential file; (c) indexed sequential file. Scaffolding: the pseudocode is given in the text, so they might just apply it to a particular data structure.

**Plenary:**
Students swap their programs with a partner who needs to create a mini-manual on how to use that program.

**Follow-up ideas/Homework**

**Extension:**
Create programs that generate random files. Students can use freely available online resources such as Microsoft, How to Work with Random Access Files

Luther Blissett, Updating a Random-Access File

Students can write a report comparing how a language like Python writes text and binary files, respectively, for example: `f=open("name.txt","wt")` and `f.write(line)` for textual files and `f=open("name.txt","wb")` and `f.seek()`

**Topic: Explain the need for security, including file backup, generations of files and transaction logs**

**Starter:**
Ask students if they have ever lost data and how long it then took them to replace the work lost. Steer the discussion into possible prevention.

**Main activity:**
A case study of a commercial organisation’s backup policies can be discussed, as well as the cost of it. An IT person working for the school can be invited as a guest speaker, if possible (or students could take a tour of the backup server room).
Here is a website for a commercial data storage centre—Telecity Group.

Or another one.

And here is the site of a company called Kroll that gets called in when the proper backups were not done.

**Plenary:**

Students are asked what they do when their computer starts running out of space. This should prompt the discussion of archiving. There could be a link to the Data Protection Act and how archiving would often have a finite life. A good question to ask would be, if students stored their old photos and videos on a DVD, how could they read it 20 years from now? How easy is it to read something from 40 years ago?

**Follow-up ideas/Homework**

Students are assigned a role of a software manager, who has two assistants whose sole job is to run backups. The software manager needs to create a manual for the assistants in a form of a report.

Students are given a sample transaction file and a sample master file. They are to write a program that merges them accordingly.

Students can research the evolution of media used for backup throughout the years, from magnetic coils and punch cards to Blu-ray and cloud storage.

**End-of-chapter answers**

1

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Size (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject name</td>
<td>String</td>
<td>10 – 20</td>
</tr>
<tr>
<td>Number of lessons per week</td>
<td>Integer</td>
<td>1 or 2</td>
</tr>
<tr>
<td>Requires ICT?</td>
<td>Boolean</td>
<td>1</td>
</tr>
</tbody>
</table>

Total the field sizes – mark awarded if sizes are added up correctly (even if they are wrong).

Multiply the total by 30

Add on 10%

2

3

**Code**

```python
function updateStock(name, amount)
    stockFile = open("stockFile", "r")
    tempFile = open("tempFile", "w")
    while not EOF(stockFile)
        tName = stockFile.read()
        tAmount = stockFile.read()
        if tName == name then
            tAmount = amount
        end if
        tempFile.write(tName)
        tempFile.write(tAmount)
    end while
    stockFile.close()
    tempFile.close()
    copy("tempFile", "stockFile")
```
Indexed files can jump to the correct group, making searching faster than sequential files.

Adding new records to indexed files requires all entries within the index to be updated making it slower than sequential files.

The location of a specific bit of data is determined by the hash function. It will give the address of the data when both adding and searching for the data. Hash functions, given the same input, will always produce the same output. However, there may be a collision meaning that data may have to be stored serially for all colliding items.
Chapter 5: Databases and distributed systems

LEARNING OBJECTIVES

• Explain what is meant by data consistency, data redundancy and data independence.
• Describe and discuss the benefits and drawbacks of relational database systems and other contemporary database systems.
• Explain what is meant by relational database organisation and data normalisation (first, second and third normal forms).
• Restructure data into third normal form.
• Explain and apply entity relationship modelling and use it to analyse simple problems.
• Describe the use of primary keys, foreign keys, and indexes.
• Describe the advantages of different users having different views of the data in a database.
• Explain how the data can be manipulated to provide the user with useful information.
• Explain and apply appropriate techniques for data validation and verification of data in databases.
• Explain the purpose of query languages.
• Construct and run queries using Structured Query Language (SQL).
• Explain the purpose of a database management system (DBMS) and data dictionaries.
• Explain what is meant by Big Data, predictive analytics, data warehousing and data mining.
• Explain that distribution can apply to both data and processing.
• Explain distributed databases and the advantages of such distribution.

What your students need to know

• Modern computers are synonymous with data.
• Big Data is a buzzword for (usually internet-based) incredibly high-volume databases such as those run by the leaders of online industries, for example, eBay, Facebook, Google, etc.
• Big Data aims to understand the needs of consumers or other trends and could be used to decode the human genome, organise traffic flows, or target for advertising. Some uses of Big Data are controversial due to invasion of privacy issues and vulnerability to data theft by hackers and governments overstepping boundaries.
• Prior to computers, data was held in paper form, which took a lot of storage and effort to copy and search. This is still occasionally the case for small businesses.
• Databases are best for keeping the data accessible and searchable.
• Many apps and websites are just interfaces to databases.
• The flat file database is a simple type of a database where all data is held in a single table. They are not suitable for large projects due to data redundancy.
• Data redundancy is a problem that not only wastes space but results in inconsistent data and failed searches (when databases are updated, we might miss some copies of the same data, for example, the customer’s name and records will look like they belong to different customers).
• Relational databases overcome the problem of redundancy through the use of multiple tables, creating another dimension in addition to the two dimensions of a flat file database. Primary and foreign keys are used to link the tables, so that a query will pull the data from multiple tables into one flat file report.
• Primary keys must be unique and most tables would have one. A key is an index and this is what indices do – they speed up searches. A primary key of one table might be linked to another table, where it appears as a foreign key.
• Secondary keys are used for faster searching, similar to tags/hashtags used in social networks and blogs, for example, ‘#LOL’. They don’t have to be unique.
• Indexing saves searching by copying out the most searched for database attributes, for example,
surname, into a separate file/table. An index table is like a table of contents or an index in a book, except instead of topics and page numbers it would have records of surnames and their location on a hard disk.

- Normalisation is the process of splitting up a busy redundant flat file database into multiple tables that form a relational database. There are various degrees of normalisation, with the ‘un-normalised’ being the lowest and third normal form being the highest, where all the data that can be sensibly isolated in its own table has been separated out.
- The first normal form is where no cells in a table contain multiple entries/bits of data and all the columns can be connected to a primary key.
- The second normal form has no partial dependencies. This means that if a table contains columns that depend on each other more than on the primary key, we can remove them to a separate table and just leave one column as a foreign key. The rule of thumb is: if multiple instances of a foreign key relate to many instances of a primary key (known as many-to-many relationship), then the table is NOT in a second normal form.
- The third normal form is about removing ‘transitive dependencies’. These are indirect links, like a ‘friend of a friend’, or a ‘student’s form tutor’s department’ where students and the form tutor’s department are linked only indirectly or ‘transitively’. We would create a separate table for our ‘friend’ and their friends will be stored there, but not in our main table.
- Third normal form databases usually feature transaction tables, for example, rather than students and courses; we will have students, enrolments (which is a transaction table as it is likely to contain more records than other tables) and courses.
- Structured Query Language (SQL) is used to manipulate most common databases. Any manipulation of data (including just retrieving it) is known as a ‘query’. We know select queries (retrieve data), insert queries (adding data to a database), update queries (change values in a database), and delete queries (delete unneeded records). Additionally, we can use SQL to create, modify and delete whole tables and their relationships.
- Queries can combine data from multiple tables, perform calculations, and use criteria to either work with all records or just the ones fitting the criteria.
- SQL is written in plain text, and is very close to English, and was created for non-computer specialists to understand. It is understood by many otherwise incompatible databases and can be combined with other programming languages such as Python, Visual Basic or PHP (a popular web server language).
- SQL has a very limited number of keywords, so it is reasonable to expect that students know all of them by heart.
- Referential integrity is important for relational databases. If a related table is updated it might not match the related data in another table, especially when foreign keys are involved. This results in ‘orphan records’. This is similar to ending with broken links on the internet if a site linked to has changed its URL.
- Successful transactions have four properties: (1) atomicity (every single objective has been achieved, every part worked); (2) consistency (data is not corrupted, database rules were followed); (3) isolation (if multiple transactions are executed in parallel the result is identical to that achieved if they were executed one after the other – so, they can’t affect each other’s operation, which might involve ‘locking’ the data for one query that a parallel query is working on); and (4) durability (changes are committed and saved to disk).
- Large databases run over multiple computers which help distribute the load, known as distributed systems.

**Common misconceptions**

- Students can often confuse the different normal forms.
- You might also want to point out that some businesses still use paper-based records as some students may not know this.
- You should also emphasise that databases don’t always look the same.
- Students may often confuse primary and foreign keys.
• Compound keys made up of two or more fields need additional explanations.
• Students might not realise that DBMS is not just a database but a set of programs that integrate and extend a database (databases).
• Students often confuse partial with transitive dependencies, and don’t really understand what a dependency is.
• They might also not understand the downsides of having many-to-many relationships.
• You should remind students about SQL syntax and how it combines with underlying languages like Python or Visual Basic.
• Students might also not know or understand the difference between entities and queries, which both look like spreadsheets on the screen.
• The concept of a database ‘view’ could be misunderstood – it could refer to a particular set of data an employee can see, or in SQL it refers to SQL procedures (pre-made queries that are triggered by SQL code).
• Programming in SQL is usually done in a host language, for example, PHP or Java. Students need to understand which are SQL variables and which are the host language’s variables.

Lesson activity suggestions

**Topic: Big Data and issues**

**Starter:**
**Demonstration of the Wayback Machine.** This site keeps a copy of many web pages going back to the 1990s. Is it possible to see what your school’s website looked like ten years ago? This could be followed up with a discussion of issues around copyright infringement, uncovering things that are best left forgotten and the techniques used to grab whole websites with pictures and data and copy them into this database.

**Main activity:**
Discuss the underlying database nature of modern apps and websites. You could ask the students to brainstorm how a games company would keep track of in-app purchases of lives, ammunition, etc. What data would need to be recorded?

Another avenue to explore is to talk about the iOS and Google mobile devices. It is known that if your mobile got stolen, you wouldn’t have to pay for your apps again; Apple or Google will restore these apps to your new device. What kind of data needs to be stored on your device by these companies to allow this?

**Plenary:**
Discussing the Snowden revelations is also a good attention getter and is related to many current political issues.

Students can be tasked with researching five famous data breaches, for example, Google’s accounts, Wikileaks, iCloud theft of celebrity photos, etc.

**Follow-up ideas/Homework**

**More with Wayback Machine:**
Give students a selection of the currently popular sites, for example, Sky or Tesco, and ask students to find the oldest and the least cool versions of these sites design- and content-wise (for example, advertising iPhone 1).

For paper-based storage, students can research pictures of an old-fashioned law offices/medical surgeries with boxes of records stored. The Office Museum website has some good illustrations.

**Topic: To understand relational database, flat file, primary key, foreign key, secondary key, normalisation and indexing**

**Starter:**
Take students through the lookup process that forms the foundation of RDBMS. Excel’s Vlookup feature can be used to create a mock-up of a relational system, where data is stored in multiple sheets and then brought together to yet another sheet using the lookup formula.

**Main activity:**
Use MS Excel, MS Access, PHP, or a general purpose language like Python to execute queries on a database. MS Excel is the best for modelling un-normalised databases quickly. Alternatively, use this site to convert a CSV file to a SQLite database and use the SQL window to run some fast commands.

**Plenary:**
Students are given databases that have errors and they are to spot the errors.

**Follow-up ideas/Homework**

Flat file databases are best illustrated with Excel (or another spreadsheet). Every sheet is a flat file database. Students could be instructed to construct an un-normalised (aka back-of-a-napkin) database by recording this chapter’s examples in Excel. Then, as the process of normalisation is explained, they will move the data to other sheets and name them accordingly. Excel will allow labelling the columns, and through the use of ‘3D’ formulas we can implement primary and foreign keys, too!

Using MS Access will be a logical step from there. If there is a capacity to use PHP or MySQL with HTML/web hosting, it might be a good exercise. Python comes with the SQLite package that allows students to implement all the queries mentioned in the chapter text.

SQLZoo is an excellent site for learning and trying things fast.

SQLFiddle is a way of sharing snippets of SQL code (could also be used for submitting and marking work using the ‘share’ feature).

Create more queries, create another database of their choosing.

**Topic: To understand normalisation to 3NF**

**Starter:**
In an MS Access database, use a query wizard to create a report or a query, then switch to the SQL view to see what MS Access did behind the scenes.

**Main activity:**
Students convert between pseudocode and SQL going through multiple examples.

For example, generating SQL for merging Master and Transaction files.

**Plenary:**
Referring to an earlier chapter on file structure, teacher asks students what format SQL files are in: serial, sequential or random.

**Follow-up ideas/Homework**

Given an MS Access database, create a Python SQLite database with identical functionality.

Use MS Excel and MS Access to normalise the flat file database. Given an MS Excel workbook, create a corresponding MS Access database.

With the same data structure, create queries in MS Access and Python, compare the SQL code that these packages generate and comment on the differences.

Students are to investigate XML, particularly the JSON standard for data storage. Create a program that can read JSON format and convert it into SQL INSERT queries, filling up a database.

**Topic: To understand referential integrity**

**Starter:**
Bring up the problem with dead links on the internet; how do they happen?

**Main activity:**
Create a copy of the MS Excel/MS Access/Python database and remove some data. Demonstrate the errors that result.

**Plenary:**
Brainstorm pseudocode for validation that can prevent data integrity errors. Link back to the earlier chapter on data security and the importance of regular backups.

**Follow-up ideas/Homework**
Research the ways that referential integrity of a database can be restored (MS Access has a database repair utility, while in Python we might be able to program a routine that updates the links restoring the integrity). A particular example would be ‘cascading deletes’.

**Topic:** To understand transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy

**Starter:**
Ask students why a Microsoft Office document creates a strange copy of itself with ‘~’ in front of its name when it is opened.

Why, if two people open the same copy of a Word document, will one of them see the message ‘This file is in use by another user. Open read-only?’

The name of the file they opened should appear with the tilde as shown above.

**Main activity:**
Get the students (once they are comfortable with creating MS Access queries) to try and create a relational database that will break these principles.

**Plenary:**
Some Big Data companies are moving away from SQL to NoSQL. Outline the main principles behind it.

**Follow-up ideas/Homework**
Most browsers can store ‘cookies’. Can these collections of cookies be considered databases? Are they SQL databases? (The answer: no, they are pair collections, otherwise known as dictionaries, which also forms the foundation of NoSQL databases.)

Twitter uses ‘hashtags’ to organise tweets. Billions of individual tweets need to be accessed almost immediately. Searching by user name or topic is available, where a tweet might contain potential search criteria preceded by ‘#’, for example, ‘#firstworldproblems’. These can be easier indexed and responses of millions of users to a topic can be organised into threads and even analysed statistically.

Is this SQL or NoSQL?

**End-of-chapter answers**

1.
All the data is stored in single table.

2.
The primary key is a unique identifier, usually created by the system, e.g. an ID number.

3.
A database made up from lots of smaller tables, linked together by primary and foreign keys.

4.
Everything from the Students table will be returned.

5.
To ensure that every foreign key relates to a primary key from another table.
Chapter 6: The operating system

LEARNING OBJECTIVES

- Describe the need for and the role of the operating system kernel in managing resources, including peripherals, processes, memory protection and backing store.
- Describe the need for and the role of the operating system in providing an interface between the user and the hardware.
- Explain the hierarchical structure of a directory and describe file attributes.
- Explain the need for and use of a range of utility software.
- 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.
- Explain the following types of system: batch, single-user (standalone), multi-user (multi-access), multi-tasking and multi-programming.
- Explain the need to design systems that are appropriate to the variety of different users at all levels and in different environments.
- Describe a range of conditions or events which could generate interrupts.
- Describe interrupt handling and the use of priorities.
- Describe the factors involved in allocating differing priorities.
- Explain the reasons for, and possible consequences of, partitioning of main memory.
- Describe methods of data transfer including the use of buffers to allow for differences in speed of devices.
- Describe buffering and explain why double buffering is used.
- Describe the principles of high-level scheduling: processor allocation, allocation of devices and the significance of job priorities.
- Explain the three basic states of a process: running, ready and blocked.
- Explain the role of time-slicing, polling and threading.

What your students need to know

- Students need to know the principal functions of a generic operating system. It is worthwhile including a brief history and evolution timeline of operating systems. They should also be aware of different categories of operating system, for example, desktop, server (Network Operating System), mobile, embedded and distributed.
- The user interface of an operating system allows the user to interact with the kernel via the shell. The interface may be command line or graphical (GUI). Increasingly, natural language interfaces are being used that allow users to enter voice commands to operate their computer.
- The file system allows for data to be stored permanently in a structured format on storage devices. The whole structure consists of directories and their sub directories. The top level directory is called the root. Beneath the root are sub directories that can be accessed by the system administrator or root user. Below this are other sub directories. Users will have access to their own home directory and others that they have been assigned permissions to access. The file allocation table (FAT) keeps track of all files and their locations on the storage media. Operating systems use different file systems, for example, Microsoft Windows uses NTFS, Linux and Unix use ext4 and Apple Mac computers HFS+. These are not compatible, although it is possible to read files written on another operating system using a system utility. File attributes indicate the permissions that each user group has to access it, the filename extension, whether or not the file is compressed or encrypted and whether or not the file has been archived.
- Utility software allows the user to maintain and manage their system, for example, security software, disc defragmenter, backup utility and configuration programs.
• Batch processing is carried out at regular intervals, for example, once a month, and involves large volumes of data. An example is payroll processing. Batch processing may be carried out using a mainframe computer with an operating system developed specifically to facilitate this. Real time control systems are often embedded. Examples of typical systems are the engine control unit (ECU) in a car. Real time transaction systems process data instantly. An example application is an airline seat booking system.

• Batch operating systems are specifically designed to handle large volumes of data periodically such as calculating income tax payable. They are often used on mainframe systems.

• A single-user or standalone operating system will be installed on a single device and cannot support multiple people logged in at the same time. An example is a tablet or smartphone operating system.

• A multi-user or multi-access operating system can permit multiple people to log in. Each will have a local account on the operating system to assign the appropriate policy for their level of access. Most desktop operating systems fall into this category.

• Multi-tasking operating systems permit multiple processes to execute at the same time. The vast majority of modern operating systems fall into this category. In order for this multi-tasking to happen, the operating system needs to be able to manage CPU scheduling and handle interrupts.

• Multi-programming operating systems are similar to multi-tasking operating systems but the processor can only execute one program at a time. Its principal aim is to use the CPU to maximum efficiency and reduce idle time.

• The human computer interface (HCI) of an operating system permits interaction with the computer system and needs to be appropriate to the intended user. For example, a mainframe computer is likely to be used by an expert so a command line interface is appropriate. Novice users may prefer a GUI such as seen in a desktop or smartphone operating systems. Increasingly, natural language interfaces are being used with voice command input.

• Interrupts force executing processes to be stopped to allow another, higher priority, process to get CPU time. Interrupts can be generated by:
  – Hardware: e.g. power button pressed.
  – Software: e.g. a program needs to run a routine (for example, to automatically save a document being edited by a user).
  – Time triggered: e.g. in data logging applications.
  – Peripheral: e.g. user has moved the mouse.

• Interrupt priority indicates how vital it is for the process to run. Priority decreases down the list above. This allows the most important processes access to the CPU precisely when needed. Lower priority interrupts may have to wait to gain access.

• In modern operating systems, main memory is allocated to fixed size page frames which can be paged out to a swap file on secondary storage memory. This is termed virtual memory and permits the system to manage many more processes than it would be able to if it just relied on main memory alone.

• Reading and writing to main memory is fast. The same operation with secondary storage memory is much slower. This would result in too much data waiting to be written to secondary storage memory. Instead, interrupts are used to allow data to be read from main memory and then written to secondary storage by using a buffer. Single buffering allows for only one operation (read or write) to take place at once. Double buffering allows for read and write operations to take place at the same time which results in more consistent data transfer.

• Scheduling is an operating system function that assigns CPU time to processes. Processes can be in three states: running, ready or blocked. The scheduler moves processes from the waiting to the ready state and places them in the ready queue. Each process in the queue will have a unique process ID and will be assigned a priority. The latter value can change to ensure that processes do not remain in the blocked state indefinitely.

• Time-slicing is a simple implementation of scheduling that allocates a fixed time interval to each process. Processes run and then are swapped out to permit another process to run which in turn is swapped when its time slice expires. In this way, all processes get an equal share of CPU time. Polling is an active process where devices are provided with an opportunity to use CPU time. Threading splits a process into multiple parallel units of execution. Multi-threading operating systems are capable of
handling threads of processes by providing logical execution pathways.

Common misconceptions

- Students can find aspects of process scheduling and memory management confusing if they do not link them with what they have previously learned about the CPU and memory. There are a number of key terms and concepts that need to be reinforced. For example, the terms multi-tasking and multi-programming are similar.
- Other potential points of confusion are the roles of buffers and interrupts. Students may be familiar with the term ‘buffering’, from attempting to view videos online, and have an awareness that a buffer provides a temporary store of data due to a mismatch between how fast the video can be processed by the computer (fast) and transmitted across the internet (slower).
- Students should strive to connect concepts to gain a holistic understanding. For example, if they understand the concept of multi-tasking and that input and output devices need access to the CPU, they will appreciate the need for and purpose of interrupts. They should also consider that some interrupts will have a higher priority than others. Interrupt handling can be drawn out as a flowchart to show the whole procedure. This will provide a helpful revision aid.
- The topic of memory allocation can be a tricky one for students. A glossary can be helpful as there are a number of key terms that they need to be clear on. It is beneficial that they understand the need for virtual memory and how double-buffering operates.
- Queuing of processes and scheduling also needs to be understood holistically. A larger group of students can role play the process (see lesson suggestion below).

Lesson activity suggestions

**Topic: Types of operating system**

**Starter:**
‘Logo quiz’ presentation of well-known and lesser-known operating systems.

**Main activity:**
Review the main functions of an operating system. Circus activity of different computer platforms, for example, desktop, games console, Raspberry Pi, mobile device (students can investigate their own), an embedded device (such as a satellite navigation device). Fact sheets with images could be used instead of devices, for example, a real time OS, distributed OS, server OS, etc.

**Plenary:**
Complete a table showing type of OS versus typical use.

**Follow-up ideas/Homework**

Students to investigate a single operating system of their choosing to present a 5–10 minute seminar about it the following lesson. It is helpful to provide suggestions so that a range can be investigated. Resources produced can be shared to develop a comprehensive overview of a range of operating systems and to keep the task current.

**Topic: Memory management**

**Starter:**
Memory ‘mix and match’ using three sets of cards covering names of types of memory, approximate quantities and function. Registers, cache, RAM, secondary storage, and ROM should be included.

**Main activity:**
In-depth look at how operating systems manage memory. Students can produce a poster illustrating how different types of memory are utilised within a computer and the process of swapping blocks between RAM and virtual memory.

**Plenary:**
Discussion of the limitations of memory management. Why is a reliance on virtual memory a performance disadvantage?

**Follow-up ideas/Homework**

Examination questions.

**Topic: Process scheduling**
Starter:
Organise students into small teams. Give them a series of tasks to complete, for example, sharpening a pencil, building a model using bricks, making a paper model, drawing a house, etc. Initially, only one person (processor) should be used. The aim is to complete as many tasks as possible in two minutes. Then allow students to use multiple processors, or to reorder the sequence of tasks. This will naturally lead to ‘multi-tasking’ and possibly even ‘multi-threading’ solutions as well as an appreciation of task queuing and the resources required.

Main activity:
Discuss key scheduling concepts. Students to work collaboratively to produce a Wiki.

Plenary:
Discuss how real time processing and batch processing would differ from algorithms based on conventional scheduling.

Follow-up ideas/Homework
Produce a Prezi or other animated presentation explaining the basics of scheduling.

Topic: Buffers and interrupts

Starter:
Discuss how to solve the problem of sending data to a slower device from a faster one. Allow the students to suggest possible solutions illustrated by a large slowly leaking bucket being filled by a smaller tea cup.

Main activity:
The concepts of interrupts and interrupt handling build upon the topics of previous lessons. Students can work in small teams to produce a set of quiz questions on the topic of buffers and interrupts. The quiz can then take place with groups asking their own questions to the rest of the group.

Plenary:
Hand out some examination questions. Allow students to talk through what they consider to be the key points to include in the answer. This could become a longer activity if small groups are given different questions.

Follow-up ideas/Homework
Investigate the history of the development of interrupts. There are many excellent sources of information available on the internet. This activity will allow students to practise their research and writing skills.

Topic: How operating systems function

Starter:
Provide the stages of booting a computer as separate cards. Students must arrange these in the correct order.

Main activity:
This lesson will focus on the relationship between the operating system, hardware and application software. Virtualisation is common practice on network servers. Students should consider the advantages of virtualisation in modern systems. If possible, a practical activity could be undertaken to install Oracle VirtualBox and a virtualised OS, for example, Ubuntu Linux.

Plenary:
Discussion of the possible future direction of operating systems.

Follow-up ideas/Homework
Investigate the differences and similarities between virtualisation and emulation. Students may have some experience of the latter and they could also prepare discussion points on the licensing implications of these technologies.

End-of-chapter answers

1

(a) Allows memory to be partitioned, through either segmentation or paging and allocated to processes. It ensures that processes cannot access each other’s segments/pages for security and stability purposes. Memory management also allows for the use of virtual memory when RAM is almost full.
(b) Scheduling will ensure that all processes get processed, taking into consideration any priorities that may exist. It will process the maximum number of processes in the least amount of time ensuring high throughput of processes. It ensures that all users of a system will get fast response times by making the most efficient use of the processor.

2

Memory is allocated to a process and recorded using a page table. Each process will have its own page table which will say where in RAM the page resides. A process will not be aware of what pages it uses, but rather will use a virtual address space. On memory access, a virtual address is converted into a real address using the page table. A process sees its memory pages as a continuous block, but each page may reside in different parts of memory.

3

When a process is suspended it will have its process control block saved into virtual memory. This will include the stack and the contents of registers (PC, ACC, etc.). A process may need to be suspended if there is a limited amount of RAM remaining and a new process has been started. The memory manager will try and choose a process to suspend that is not currently being used. Once a process has been suspended it must be loaded back into RAM in order to continue processing.

4

Data will be sent to the hard drive and stored into the hard drive’s buffer. As the buffer works much faster than the hard drive itself, the CPU will be free to run other processes while the data is saved and the process which is saving the file becomes blocked. Once the buffer is full, the hard drive will slowly save that data. When the buffer is empty and the data saved, an interrupt is sent back to the CPU. On the next FDE cycle, the interrupt register will be checked and, depending on priorities, the blocked process will be brought back into the ready to run queue and more data sent to the buffer. This algorithm repeats until the file is fully saved.

5

(a) Compression software will reduce the size of a file and may also archive a group of files into a single file. When a file is to be accessed it must be decompressed first.

(b) Anti-virus software will use heuristic methods to match known virus signatures or behaviours to what is happening on the computer. If a virus is found, it will be placed into quarantine. Each file on the system will have to be scanned.
Chapter 7: The need for different types of software systems and their attributes

Learning Objectives

- Explain the use of a range of types of software, including open source software, bespoke and off-the-shelf.
- Explain that some computer applications are safety related and require a high level of dependability, and hence that the development of safety critical systems is a highly specialised field.
- Describe the role of the computer in weather forecasting, computer aided design, robotics and the use of computer generated graphics and animation.
- State the nature and scope of computer control and automation.
- Describe the benefits and implications of automation.
- Explain the purpose, use and significance of expert systems.
- Discuss the possible effects of expert systems on professional groups and the wider community.
- Describe the use of search engines on the internet.
- Describe common contemporary applications.
- Discuss the possible effects of the internet upon professional groups and the wider community.

What your students need to know

- Open source software is distributed with the source code and a licence which allows the software to be modified and distributed by the user. It is usually developed and supported by a community. Closed source software is usually developed by a commercial organisation and protected by a licence which forbids reverse engineering or distribution of unlicensed copies. It may be developed to meet a specific brief for a client, termed bespoke software. Off-the-shelf software is generic and will meet a wide range of client needs. An example of this latter type would be a generic spreadsheet package. Off-the-shelf software may be customisable to enable clients to set up their software to meet specific needs, for example, setting up a spreadsheet template for a customer’s account details.

- Bespoke software is often required to meet very specific needs. In industries where software is developed for safety critical systems, it is imperative that the project is checked carefully at each stage of the system development lifecycle. This ensures that the requirements have been correctly defined, that the design is accurate and that software meets the required level of performance and is free from errors. The process of validation checks that the correct piece of software is being built, i.e. that it meets the requirements. Verification confirms that the software functions correctly with no errors. It must accept only the correct inputs, must perform the correct operations and produce accurate outputs. The software must perform consistently when given the same inputs. These processes are part of the formal quality assurance of the product. An independent body may be used to carry out validation and verification to ensure that reviews are objective.

- Expert systems are designed to carry out the decisions usually made by a human expert. They are often used for diagnostic systems. In a medical diagnosis system, expert doctors provide facts about particular conditions to build a knowledge base. The system will also have a user interface and an inference engine for handling queries. The links between facts are called rules and are stored in a rule base. Expert systems belong to the branch of computing termed artificial intelligence. Expert systems can only make decisions that have been entered into the knowledge base; however, more sophisticated systems can add new facts and rules, for example, those entered by the user. Expert systems can be used by non-specialists who may not be able to distinguish between a real and erroneous diagnosis. For example, feeling lethargic, having a raised temperature and skin rash may be symptoms of a number of different conditions. Online medical diagnosis utilities can cause a great deal of worry to patients. A car fault diagnosis system can read sensor data and provide an accurate fault diagnosis in a very short space of time which may save the owner time and money.

- Application software or ‘apps’ permit the user to carry out useful tasks. Examples of applications are:
word processor, desktop publishing, presentation software, spreadsheet package, database software, a game, a calculator program. Utility software allows the user to maintain or configure their computer system. Examples of software utilities include: file manager/explorer, disc scanner and clean up, system configuration, security, backup and archive, compression software, disc defragmenter and encryption software. Mobile computing devices such as smartphones and tablet computers, will link to an online app store where applications and utilities can be downloaded onto the device.

- Computer applications have allowed everyday tasks to be performed quickly and sometimes automatically. Computers and smartphones contain a calculator program which reduces the requirement for users to carry out calculations manually. Information can be searched for and results returned in an instant. This in turn is increasing our reliance upon computers and the internet. Students should be able to discuss advantages and disadvantages of these ideas.

- Modern creative industries rely on a combination of traditional video and computer generated imagery (CGI) and animation. Traditional hand-drawn animation was a very time-consuming process. Specialist software, such as Adobe Flash, has enabled even novice users to produce computer animations. 2D animated screen objects are referred to as sprites. 3D animation is more complex. Software is used to create wire-frame models from simple polygons. These are then rendered by the software to make them appear more realistic. Examples of 3D modelling software include Blender.

- Specialist software is used extensively in the field of weather forecasting. Software used at weather stations must be capable of data logging. The data is then analysed and, based on previous scenarios, a simulated model is generated of the most likely weather pattern. Data patterns can be analysed to increase the accuracy of future predictions and to extract long-term trends.

- Computer control systems use sensors to provide an input signal which is then processed by software to generate an output signal. The output signal can be used as feedback in a closed loop control system to enable the output to be turned on and off automatically, for example, a thermostat in a heating system. The software used in these systems is usually quite simple as the amount of data being processed is small and decisions made very simple, i.e. turning a heater on or off. Automation uses computers to carry out processes without the need for human interaction. Software must have a high level of reliability and robustness. It is also important that systems are calibrated to provide the correct degree of accuracy.

- Automated systems are expensive to set up but may replace the work of several employees so can save money in the long term. Some tasks are difficult to automate as they require very fine movements that are difficult for machines to replicate. Robots can be used to work in environments that are dangerous for humans such as mines, under the sea or in nuclear power station cores. Robots do not require rest breaks but, as they become more capable and are given greater responsibilities, it is important to debate whether robots have rights. Robots can also perform operations on humans, the ethics of which can form the basis of a very interesting discussion-based lesson.

- Robotics is a specialist field of computing and automation. A robot is a device that carries out automated tasks. Robots are used in the manufacture of cars to assemble components, weld and paint panels. The software used to control robots provides instructions needed to carry out specific actions. Programs need to be carefully tested as they will often run unattended. Humanoid robots, such as Honda’s Asimo, are designed to mimic human movement. They contain sensors to enable the robot to make decisions. For example, when Asimo puts a foot forward, pressure sensors will feed back readings to indicate when contact has been made with the floor. This will then trigger the second foot to be lifted as Asimo walks. More details of Asimo can be found here.

- Computer aided design (CAD) software is often used in conjunction with computer aided manufacturing (CAM) software. CAD allows a user to design their product using a computer as 2D views from which it can produce a 3D view. CAM software will then convert the design into a 3D product. Examples of devices that use CAM are 3D printers, laser cutters and CNC lathes. Many household products are produced in large volumes using automated systems. This reduces production costs and increases output.

- Search engines are programs for locating web pages containing key words entered by the user. Web pages also contain meta tags which may allow a web page to appear more highly ranked in search results. The algorithm used by a search engine is very important in returning the most relevant results rather than a large number of pages. Google’s search engine uses the PageRank algorithm which uses the number of links to a page as a measure of its importance.
• Search engines store web pages gathered by a web crawler or spider. The page data is then indexed to facilitate fast searching. Search terms entered by the user form the search query which is used to search the index for matching terms. The pages are then returned in the user’s browser as the result set.

• Professional organisations have a presence on the internet which may enable them to reach a larger audience. Websites can be used to create professional profiles which may assist in career development. Journals are made available online and learning communities allow students to study remotely.

Common misconceptions

• The topic of expert systems is one with specific terminology which students need to understand. They can confuse the terms fact, rule and query so these need to be accurately learned. It is also helpful to look at examples of expert systems such as the NHS Choices assessment tool. Students can consider common conditions such as a sore throat and run through the symptom checker.

• Search engine operation can also cause confusion. Students should be clear about how the search engine builds the index of pages and handles search queries. It is worthwhile investigating different search engines and the algorithms that they use so students have a better understanding of how relevance is maximised.

Lesson activity suggestions

Topic: Nature and variety of application software

Starter:
‘Mix and Match’ activity of common applications and their uses. For the starter, confine it to software that students will have had experience of using, i.e. word processor, desktop publishing, graphics package, email client, etc.

Main activity:
Students should consider the facts and rules about a scenario, for example, diagnosing computer faults or identifying animals. They can design a user interface and consider how the system will function, i.e. what queries (goals) the user may wish to solve.

Provide examples of different sectors and then students will need to consider what software applications would be most appropriate. There are a number of online videos showing how large organisations use software applications.

Investigate current most popular mobile apps and describe priorities when developing software for mobile devices.

Raspberry Pi activity:
Install the Libre Office software package and evaluate its features.

Plenary:
Use sticky notes to record new facts that students have learned in the lesson and one question that they wish to have answered.

Follow-up ideas/Homework

Examination questions.

Topic: Control systems

Starter:
Define control system terminology: sensor, input signal, output signal, feedback.

Main activity:
Design a control system on paper for an automated greenhouse. This will control the temperature, amount of shade to prevent scorching of the plants and watering of the plants. They should consider what sensors and output devices are required and develop an algorithm for the system. If software is available such as Flowol, this can be used but it is important that students also gain experience of developing pseudocode.

Plenary:
Consider how their greenhouse system could be made more reliable. What safety systems will be put in place?
Raspberry Pi activity:
There are a wide range of simple ideas for creating control systems using the Raspberry Pi. An interface board such as Piface Digital can be added to provide ample input connections for sensors and output connections to buzzers, LEDs and motors (via the two relays). More information can be found here. Additional ideas and components can be found here and Simon Monk’s site. Students could create a simple control system to turn on a fan when a temperature sensor detects an increase in value.

Follow-up ideas/Homework
Research the use of control systems in a modern car and produce an annotated poster showing these.

Topic: Implications of automation and expert systems

Starter:
Define these terms: expert system, artificial intelligence, robot and control system. Students can investigate typical applications.

Main activity:
Create a set of scenarios on cards. Scenarios should be realistic and pose ethical questions, for example, a skilled welder is going to be replaced in a factory by an industrial robot. What help should be provided to the displaced employee? A computer control system fails and causes an accident. Who is at fault? If you are chatting online and you find out that the person that you are talking to is a program, is this ethical? Can robots commit crimes? How do you feel about getting into a car with no drive? Is it ethical to switch off a humanoid robot? Divide the class into groups of up to four students and give each a scenario card. Give the group five minutes to discuss the card. They will then reconvene as a whole class and each group will describe their scenario and what they discussed in under five minutes.

Allow at least 20 minutes for this section as it is important for students to explore the issues fully.

Show students this video about the rise of artificial intelligence.

Put students back into small groups to answer questions posed by the video: should we be concerned about the increased use of artificial intelligence? How can expert systems, for example, those used in diagnostics, have a negative impact? (Revisit the NHS Choices self-diagnosis tool. Is it possible for a patient to misdiagnose their condition?)

Plenary:
Provide students with news clippings of current stories related to automation and expert systems, for example, self-driving cars, job losses due to automation replacing workers, Professor Stephen Hawking’s concerns about artificial intelligence. Students summarise the benefits and drawbacks of these technologies as bullet points.

Follow-up ideas/Homework
Create a simple expert system for identifying items from a chosen topic, for example, games characters, animals, football teams, flowers, etc. Using a programming language of their choice, construct input statements to read in user input, select case statements to determine the appropriate output. The program should then process the user’s answer through another select case construct to pose another question until the item can be correctly identified.

Produce a mock newspaper article about increased use of artificial intelligence in software.

End-of-chapter answers

1
Open source software will come with the source code while free software will not. With open source software you are allowed to alter, update or add to the source and then republish it.

2
Bespoke software is designed and built for an individual, small group or business. Off-the-shelf is more generic and is designed to work with a wider range of uses and needs. Bespoke software will be tailored to the specific needs of the user. With off-the-shelf software the user may have to alter the way they work to fit in with how the software works.

3
Knowledge base – a store of all of the facts for use by the knowledge base system.

Rule base – links facts together to help arrive at solutions.

Inference engine – will decide on what question to ask next and if we have enough information to determine the answer.

PageRank will look for hyperlinks on every page it indexes. The more links that point to a single page (from other websites) the higher that page will be in the search results.
What your students need to know

- Data security and integrity are important and breaches are common. Students need to know the list of common threats, from black hat hackers to employees leaving disks on a train. There are industry practices and techniques that minimise the chances of losing data or getting it into wrong hands, which includes cryptography, clerical practices, write-protection, biometric authentication, etc. Yet, none of them are 100% secure and contain multiple weaknesses.
- Data damage can result from inadvertently creating copies of data and letting different departments use different copies, so that the data integrity is lost.
- Computer security can be compromised in a non-computer way, such as through not shredding office documentation and memos, so that somebody could go through the waste bin and obtain passwords and security protocols.
- Security is a multi-level process which brings together physical security of computer rooms, documentations, password access, network protection, etc.
- Students need to understand the difference between different cryptographic methods and their relative strengths. They also need to know that no perfect protection exists, so they might be asked to write a mini-essay on the importance of contingency planning BEFORE a disaster happens.
- Encryption scrambles plain text into cipher text which can’t be understood. Decryption is the process of unscrambling the encrypted message to bring back the original. The piece of information needed to decrypt the cipher text (which itself needs to be kept secure) is called the key.
- Ciphers have been around for a long time. Caesar’s cipher is simple and famous: every letter in an original message is replaced with the one either a few positions ahead or behind it in an alphabet. This can easily be broken by modern computers by trying all possible combinations.
- Encryption is now used in almost all communication and the power of computers is making older encryption methods unsafe. Alan Turing was the first to use a computer to break enemy codes, which he did successfully during WWII.
- In symmetric encryption, decryption is simply the opposite of the encryption process and the same
key is used for both. It is easy to do but is also easily cracked by the brute force of modern computers.

- In asymmetric encryption, otherwise known as ‘public key encryption’, the encryption and decryption keys are different, so the party that encrypted the message can’t decrypt it and the party that can decrypt the message can’t encrypt it. It is much more secure but was unknown until recently. In this method, the encryption key is public and can be freely shared by the party that needs to receive an encrypted communication. The decryption key is private and secret, though. Anybody can find a public key of another person and use it to send them an encrypted message which only the recipient can decrypt.

- This is achieved via a mathematical one-way function that can only be reversed under a single circumstance. Such functions are called RSA ciphers (from the three inventors’ initials). The function requires a person to choose three large prime numbers, of which one and the product of the other two are published collectively as a public key, for everybody to encrypt messages to this person. An exponential function with modular division is applied to every ASCII character in the message. The original recipient is the only one who knows the three secret numbers and there are too many combinations of the products of these numbers for somebody to guess them.

- While secure, public key encryption is vulnerable to corruption and interference with the message. Hashing helps with that. A hash is a unique string of data through which the original message is put. Unlike a key, which is unique to a person but is the same for all the messages from/to this person, a hash is unique for every message and is sent along with the encrypted message to the recipient. Any smallest change in the original message would generate a completely new hash. Upon receiving the message and the hash, the recipient decrypts the message, generates a hash and if whatever has been generated doesn’t match the hash that came along with a message, the message has been tampered with! Hashes are secure; they can’t be reversed without trying an infeasible number of possible combinations. So, a hash is a bit like a unique signature.

- Another approach to protecting data security is the use of biometrics. There are a few ways to use unique characteristics of a human body to generate biometric authentication and students should be able to suggest the most relevant ones given a scenario. Biometric authentication simplifies user experience with no passwords to remember and can speed up logins, however, ageing and loss of body parts can make the system not work for everyone.

- Ultimately, security comes down to people: both those with legitimate access and those trying to gain access to sensitive information.

- Some damage is caused by employees, either on purpose or not. This can be reduced through training and the use of Codes of Conduct.

- There are multiple types of hackers, not all of them bad. There has been a fair number of high profile cases involving famous companies and governments targeted by hackers, usually to steal confidential information for either commercial or political reasons.

**Common misconceptions**

- Hackers don’t erase or disable victims’ computers, they make money by obtaining information or access to computers which they can resell.

- In practical work, students may not be able to do encryption reliably, and in programming it, they may not be familiar with string manipulation and nested loops.

- Students also confuse public and private keys (which one is used for encryption, and which for decryption).

- They may also not know what modular division or exponential functions are, which may need clarification in class.

- Finger scanners can’t be fooled by a picture of a finger: they use electrical capacitance to read the skin patterns.

**Lesson activity suggestions**

**Topic:** 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

**Starter:**
Look at the story of a lost iPhone prototype: [Gizmodo How Apple Lost the iPhone 4](#).

**Main activity:**
An overview of practices that aim to prevent unauthorised access to computers and software in an organisational setting.

**Plenary:**
A discussion of what makes a secure password. A popular blog, Splashdata, maintains a list of worst (but popular) passwords [here](#).

**Follow-up ideas/Homework**
A good number of employees working with sensitive data get special keyfobs (known as security tokens) with screens which generate new pins at a regular interval, for example, once a day or hour. Students plan and code a program that recreates this process. An algorithm could include concatenation of the hash of the date and that of a non-dictionary word, or there could be a list of passwords, for example, a list of 365 randomly generated pins. The program will display the password, the position of which on the list corresponds to the day's position in the year, for example, 2nd January 2 will retrieve the second pin on the list. Other, more secure algorithms exist and could be explored. Once implemented, students explain why they chose a particular technique and how secure this would be in a real world setting.

Another practical idea would be to create pseudocode (and code) for a program that generates secure passwords.

**Topic: Practical use of decryption**

**Starter:**
Demonstrate encryption using one of the tools mentioned in the chapter.

**Main activity:**
Designate students as ‘spies’ and give them encrypted portions of a text and their own keys. They have to decrypt their part, find other students and their portions of the text and put all the text together into an original document.

**Plenary:**
Assume that, given enough time and computer power, most encryption can be broken. Are longer messages or shorter messages easier to crack?

**Follow-up ideas/Homework**
An extension of this could be trying to do a simple example on public key encryption, similar to the one given in this chapter.

It might be fun to get one of the students to encrypt a short message using the Caesar cypher, and tell others that the offset (the number of alphabet positions all letters were shifted by) was within a certain range. Then the others, working separately, will try the brute-force approach of trying all combinations until the message makes sense.

Create a cipher encryption/decryption program in their choice of programming language (even Scratch if need be). Create a program to crack a simple cipher by trying all combinations until a certain word is recognised.

**Topic: Biometrics**

**Starter:**
Ask students if any of them had a passport picture taken. Why are they not allowed to smile or cover part of the face? Additionally, the class can look at the rules for passport pictures here: [UK Passport Office Guidance for photographers](#).

**Main activity:**
Students are given scenarios and they need to justify the best biometric technique (or argue against any biometric use in this particular case).

**Plenary:**
Students are given a sheet with a few photographs that contain identically sized pictures of faces of a number of people (alternatively, they are given the collage as an image which can be opened in a graphics editing package or even a presentation package), each face at least twice. Using the techniques described
in the text, draw the lines connecting the important facial recognition points and then transfer to a separate sheet/file. Comment on the reliability of this method of authentication.

**Follow-up ideas/Homework**

There is a view that professional cleaners and other people that deal with abrasive chemicals suffer from weak or non-existent fingerprints. Investigate if it is an urban legend or the truth.

Does the same apply to the contact lens wearers when it comes to iris recognition?

**Topic: Hacking**

**Starter:**
Ask students to play a game of associations where the teacher calls out various computer security related words. The teacher then tries to extract more than one association, for example, if ‘hacker’ is associated with ‘theft’, the teacher can introduce the concept of ‘white hat hackers’.

**Main activity:**
Discussion of various types of malware and the types of victim that each of them targets.

Are some operating systems targeted more than others?

**Plenary:**
Students are asked what they do to stay secure and what’s the worst that a hacker can do to them. This can be connected with cyberbullying.

Debate if hacking is less of a crime than physical violence or theft.

**Follow-up ideas/Homework**

Prepare a report about one hacker (or group of hackers) that got caught – what’s their background, how did they do it, what was the punishment. Prepare another report about a case where nobody was caught.

One famous hacker **Aaron Swartz** committed suicide but he had a lot of fans and was himself perceived as a **victim**. Debate this case.

**End-of-chapter answers**

1

Passwords – to prevent unauthorised people accessing the file.

Access rights – ensuring that only authorised employees can access the file.

2

Shift ciphers will exchange one letter for another using an algorithm. They do not diffuse the original plain text and as such they can be easily broken by using frequency analysis. Because they are so easily broken they cannot be used for banking otherwise people’s financial details could be easily accessed.

3

The lighting of the subject is very important so if the face or eye is not illuminated enough then the system will not be able to take accurate measurements. People, over time, may change their facial appearance. For example, plastic surgery or even growing a beard, both of which could throw the biometrics. If the person does not look straight ahead then the system would not be able to take accurate readings. Also, some people are worried about their privacy and may not be comfortable having biometric data stored about them.

4

Backup of systems – key data and files will be backed up on a regular basis.

Redundant servers – key servers will be duplicated so if one fails the other can take over immediately.

5

They can ensure that key files have access levels so only certain people can move or delete them. Staff can be trained on how to manage system resources to minimise the chances of accidents occurring. Finally files can be backed up so that if accidents do happen the files can be recovered.
Acknowledgements

Cover © 2013 Fabian Oefner www.FabianOefner.com