Edexcel

OL IGCSE

Computers

CODE: (4CP0)

Unit 04

THE INPUT–PROCESS–OUTPUT MODEL

How many computers do you have in your house? There are probably a lot more than you think. Let's find out why.

▲ How many of these things do you think of as computers?

Chapter 16 – Machines and computational models

You might be surprised to know that every device shown on page 160 contains an embedded computer. A computer is a machine that takes some kind of **input** from its surroundings, **processes** the input according to given rules, and provides some kind of **output**.

▶ **Figure 4.1** A simple input–process–output diagram

The clever bit in computer science is the activity that goes on between the inputs and outputs - the processing. Processing means performing a series of actions on the inputs, according to a given set of rules. You might have met this idea in maths already if you have studied **'function machines'**.

SUBJECT VOCABULARY

input to enter data into a computer

process to change the meaning or format of some data

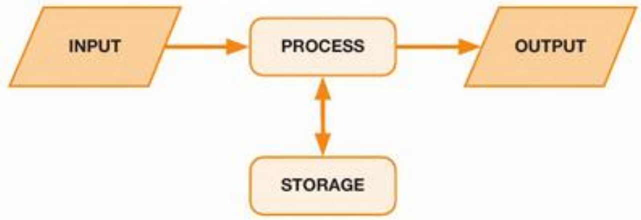
output to display or output data that has been processed (or has been stored)

function machine a metaphor or diagram that represents a machine that takes an input. It applies a rule such as a set of operations and delivers the answer as an output

▶ **Figure 4.2** A simple maths function

▶ **Figure 4.2** A simple maths function

▶ Figure 4.3 A simple function machine with storage

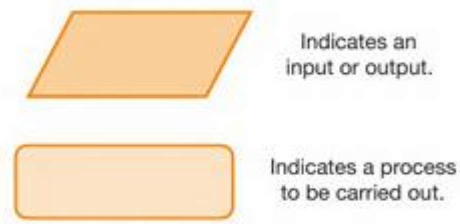


Ranges of computational models

Computational models describe how algorithms are executed by a machine, e.g. by processing data using inputs and outputs. There are different models of how this processing can be carried out.

SEQUENTIAL

In the sequential model, this is the way that we commonly imagine how computers work. This involves following instructions in an algorithm step by step, in order, from start to finish.



▲ Figure 4.4 Flowchart symbols for input/ output and process

PARALLEL

In the parallel model, computer processes are distributed between two or more processors in a computer with two or more processors installed. Each separate part of the algorithm that each processor processes can be combined together. It requires an operating system capable of supporting two or more processors. It also requires software programs capable of distributing processes between them equally.

SUBJECT VOCABULARY

agent a computer system that can interpret its environment (through its sensors). It can act autonomously upon that environment (through its effectors)

autonomous self-directed and acts without requiring any help or guidance. Can choose its own goal and uses its experience to achieve it

effector something that produces an effect by carrying out an action

MULTI-AGENT

In the parallel model, one task is processed by several processors while in the multi-agent model, separate tasks or algorithms are processed by different systems (**agents**) to perform a particular function. Each agent acts independently and under its own control. Each agent is **autonomous**, but they cooperate with each other through negotiation and coordination.

SUBJECT VOCABULARY

von Neumann architecture computer system design in which the program is stored in memory with the data

Chapter 17 – Hardware

The first computers, like your calculator, had to be redesigned - or at least rewired to do a different job. The idea of creating a general-purpose computer developed in the 1930s and 1940s through the work of Alan Turing and John von Neumann.

They proposed that the instructions for the processing could be held in storage with the input data.
A computer in which the processing instructions are stored in memory with the data is called a stored-program computer or a **von Neumann architecture computer.**

SUBJECT VOCABULARY

central processing unit (CPU) hardware device that carries out the processing in a computer

main memory/random-access memory (RAM) a temporary store for data and instructions (programs)

bus a group of connections between devices in a computer

▶ Figure 4.5 Buses used in a computer

In a von Neumann architecture computer, the hardware device that does the processing is called the **central processing unit (CPU)** and the storage is called **main memory/random-access memory (RAM)**. They are connected to each other, and to the input and output (I/O) devices, by a group of connecting wires called a **bus**.

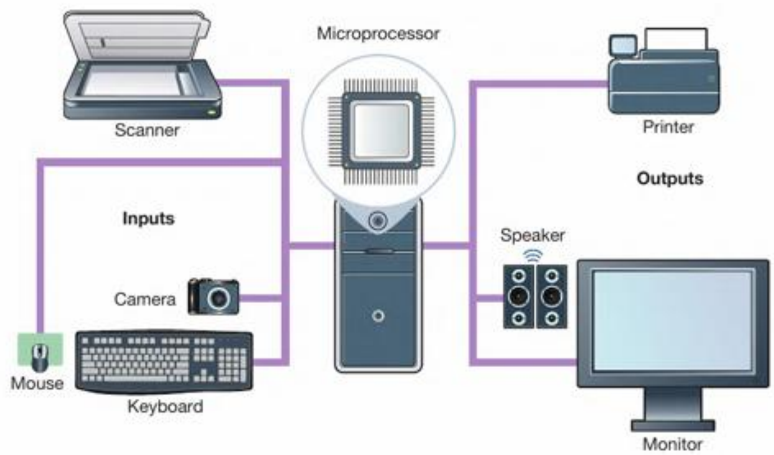
▶ Figure 4.5 Buses used in a computer

When a computer is turned on, the CPU fetches an instruction from the memory, carries out the instruction it receives, and then fetches the next, and so on. This sequence is called **the fetch-decode-execute cycle,** and it carries on until the power is turned off.

Input and output devices

Input devices such as keyboards, mice and light sensors provide the computer with data it can process. Output devices such as monitors, printers and speakers are used by the computer to communicate the results of the processing.

▶ Figure 4.6 A computer system with some common input and output devices



Hardware components of a computer system

RAM AND ROM

If you have ever bought a desktop or laptop computer, you will have seen the main memory referred to as random-access memory (RAM). This is the temporary store that the CPU uses for data and instructions (programs).

When the CPU saves data into memory, this is called **writing** - the CPU uses the bus to tell the memory what data to save and where in the memory to save it. The reverse process is called **reading**, and the CPU must specify which part of the memory to read from. Each memory location has a unique memory address, in the same way that your home is identified by an address. You can think of memory like a series of letter boxes each with space for 1 byte of data. The **memory addresses** are just numbers, starting from 0, labelling each mailbox.

SUBJECT VOCABULARY

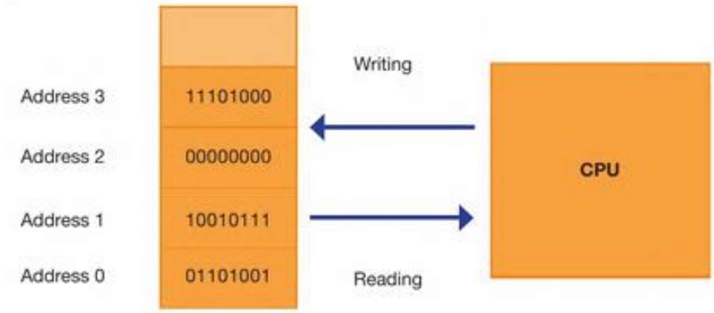
fetch–decode–execute cycle sequence of steps carried out repeatedly by a CPU

writing when the CPU sends data to memory to be stored at a given address

reading when the CPU retrieves the data stored at a given address

memory address a number that uniquely identifies a (memory) storage location

▶ Figure 4.7 Memory addressing



RAM is described as **volatile**, which means that its contents are lost when the power is turned off. Because of this, computers also need **non-volatile** memory to store any programs that must run when the computer is first turned on this memory is called **read-only memory (ROM).**

The photograph below on the left shows the main circuit board of a 1980s computer (the Sinclair ZX81). The ZX81 had either one or two RAM chips. Two are shown in the diagram, along with three other chips: CPU, ROM and a fourth chip that connects the input and output (I/O) devices. You can also see groups of parallel silver 'tracks'; these are the buses (the wires connecting the chips together).

SUBJECT VOCABULARY

volatile memory that is erased when the power is turned off

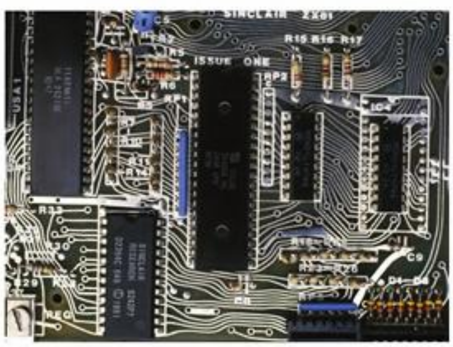
non-volatile memory that is not lost when the power is turned off

read-only memory (ROM) memory that cannot be altered and is not lost when the power is turned off

cache memory memory used to make up for the difference in speed between two internal components

Most computers require a third kind of main memory: this is called cache memory (pronounced like 'cash'). Cache memory is a small amount of fast, expensive memory that is used between two devices that communicate at different speeds, most often the CPU and RAM.

▶ Computer circuit boards: Left – Sinclair ZX81; Right – Raspberry Pi



Cashe memory

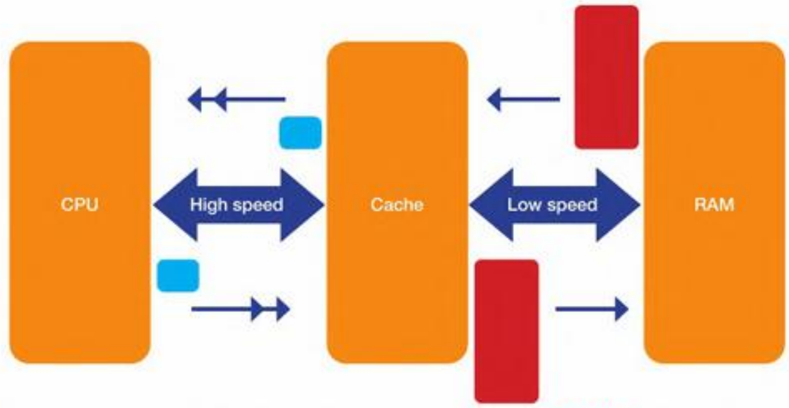
Most computers require a third kind of main memory: this is called **cache memory** (pronounced like 'cash'). Cache memory is a small amount of fast, expensive memory that is used between two devices that communicate at different speeds, most often the CPU and RAM.

SUBJECT VOCABULARY

cache miss when the data requested for processing by a component or application is not found in the cache memory

microprocessor the central unit that executes and manages the instructions passed to it

▶ **Figure 4.8** Diagram of cache memory



In a CPU cache, frequently used code or data is loaded in **chunks** from the slower RAM into the cache. The CPU accesses the cache memory at its own, faster speed. This means the CPU isn't slowed down by having to wait for data from the RAM.

GENERAL VOCABULARY

chunk quite a large amount of something

bottleneck where something is working at full capacity and cannot handle further demand

rate a measurement of how quickly something happens

Virtual memory

When a computer is running the operating system and several applications at the same time, the RAM often becomes full. Instead of closing some programs, the 'memory manager' program of the operating system will use 'pretend' or virtual memory to store some of the data, usually on the hard disk drive.
It works like this:

■ A process running on the computer may need to store data in the physical memory.

■ If there is no free memory, the memory manager will 'swap out' some of the data stored in RAM to the swap area on the hard disk drive. Then, it will 'swap in' the requested data into the now free area.

■ Usually, the least recently used stored data is swapped out.

If data is swapped out and then is needed again, it is swapped back in from the swap area, at the expense of other data.
There are disadvantages to using virtual memory:

■ The read/write speed of a hard drive is much slower than RAM, and the technology of a hard drive is not made for accessing small pieces of data at a time.

■If the system has to rely too heavily on virtual memory, there will be a significant drop in performance.

■ Often the operating system has to constantly swap information back and forth between RAM and the hard disk drive, which operates all of the time. You can hear the disk drive operating continuously. This is called **'disk thrashing'** and it slows down the execution of the programs.

SUBJECT VOCABULARY

disk thrashing a very high rate of hard disk access

Fetch – decode- execute in detail

There are many different kinds of CPU, but they share some features in common. To understand the fetch-decode-execute cycle in more detail, you need to know a bit more about what goes on inside the CPU.

The part of the CPU that does calculations and logic operations is called the **arithmetic/logic unit (ALU).**

Inside the CPU are a number of memory locations called **registers**.

Some of the registers with a specific role that are found in most computers include the following.

The steps in the cycle are controlled by a **control unit** and synchronised by an electronic clock. You have probably seen the advertised **clock** speeds of CPUs; for example, a 2.2 **gigahertz (**GHz) CPU means that its clock 'ticks' 2200 million times per second.

SUBJECT VOCABULARY

arithmetic/logic unit (ALU) the part of the CPU that performs calculations and logic operations

register a storage location inside the CPU used to hold an instruction, an address or other single item of data

control unit the part of the CPU that organises the actions of the other parts of the CPU

clock an electronic device inside a CPU that 'ticks' at regular intervals and is used to synchronise the actions of the other parts of the CPU

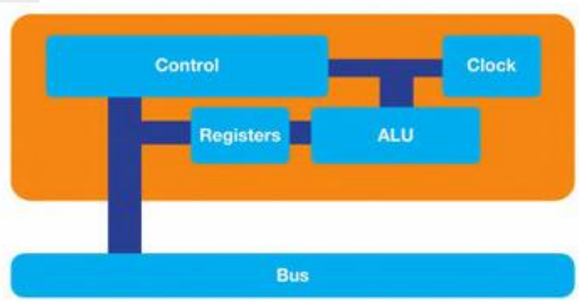
gigahertz (GHz) a measure of frequency equivalent to 1000 million cycles per second

bus width the number of wires that make up a bus. This determines the range of binary numbers that can be communicated

▶ **Figure 4.9** Clock signal

The bus that connects the CPU to other devices in the computer is split into three parts: the address bus, the data bus and the control bus.

▶ **Figure 4.10** Inside a CPU



The number of connections on a bus is called the bus **width**. Since each connection represents a binary digit (a 1 or a 0), a greater bus width means larger number values can be communicated.

Factors that affect CPU performance

Everyone wants their computers to work faster and faster. Manufacturers have continued to increase the speeds at which computers work, but there are many factors that affect CPU performance.

CLOCK SPEED

The rate at which instructions are processed by the CPU is controlled by the clock speed. The faster the clock speed, the faster the rate of processing. Or, to put it another way, a new instruction will start processing on a clock tick: the more clock ticks there are, the faster the instructions will be processed. Clock speeds have increased, and a rate of 3 GHz is common in modern computer processors. However, increasing the clock speed to increase processing speed has disadvantages:

■The instructions are processed by transistors and the rate at which they operate is limited.

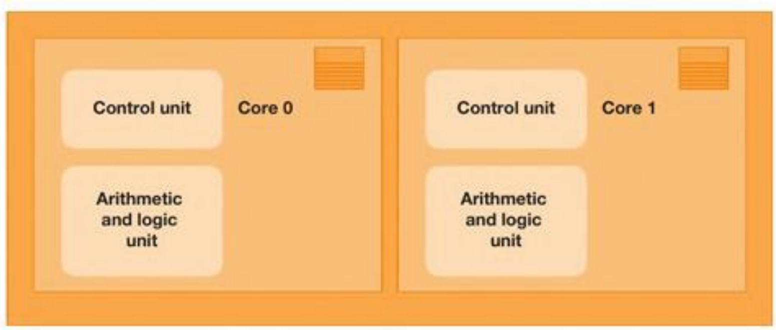
■The processor generates a large amount of heat and this increases as the clock speed increases. A fan and heat sink help to get rid of excess heat and prevent it from breaking or even melting. However, there are limits to the rate of cooling.

■Processors with clock speeds of 9 GHz require cooling by liquid nitrogen.

NUMBER OF PROCESSOR CORES

Manufacturers introduced **multicore processors** in 2006 to increase processing speed. A multicore processor has more than one processor core (but only one CPU). The following diagram illustrates the structure of a **dual-core processor.**

▶ Figure 4.11 The structure of a dual-core processor



The following table shows the names given to processors having different numbers of cores.

NUMBER OF CORES	COMMON NAME
1	Single-core
2	Dual-core
4	Quad-core
5	Penta-core
8	Octa(o)-core
10	Deca-core

The advantages of multicore processors over single-core processors are that the cores can work:

- together on the same program - this is called parallel processing
- on different programs at the same time - this is called multitasking.

However, not all programs will run at twice the speed with a dual-core processor. The tasks required might not be able to be carried out in parallel.

SIZE OF CACHE

Bottlenecks occur when one component cannot work as fast as other components and so slows down progress. In the fetch-execute cycle, this bottleneck is caused by the main memory that is used to store the instructions and data.

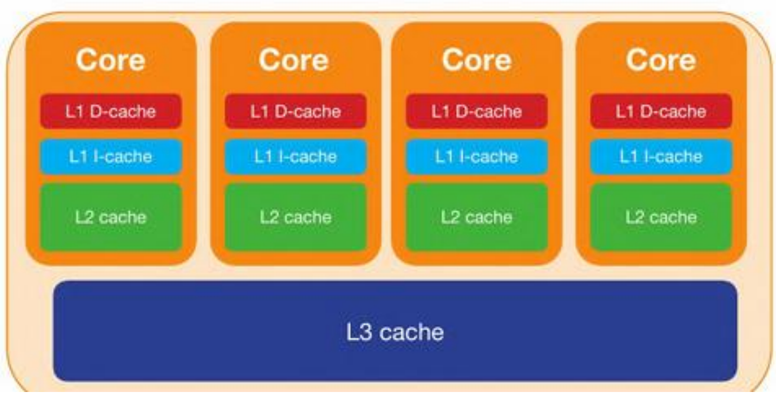
The solution to this bottleneck problem is to use faster memory very close to, or even within, the CPU. This memory is used to store recently used data and data likely to be frequently used (which is called a **cache).**

The use of caches allows the CPU to check the fast cache for the data it needs. It does not have to wait for it to be fetched from the slower main memory. The faster **static RAM** (**SRAM**) is used for the cache.

The caches are located on the processor chip. The fastest is the Level 1 cache and is smaller than the Level 2 and Level 3 caches.

The sizes and positions of the caches are shown in the following diagram.

▶ **Figure 4.12** The size and positions of caches



Secondary storage

In most situations, you need to be able to store your data and programs after the power is turned off. You have learned that RAM is volatile. This means most computers need to be able to copy the contents of their RAM to another kind of storage that is not volatile - a type that doesn't lose its contents when there is no power. This more permanent storage is called **secondary storage.**

Magnetic storage uses the fact that magnets have north and south poles. By making something behave like a magnet, the north and south poles can represent the 1s and Os of your data. This is used in hard disks and magnetic tape storage.

Optical storage is used by CDs and DVDs. Shinier or more reflective parts of the disk represent the 1s or Os.

Solid-state storage or 'flash' memory (such as USB memory sticks or SD cards) represents the 1s and Os with little pools of trapped electrons on a microchip.

MAGNETIC SECONDARY STORAGE - HARD DISKS

Inside a hard disk drive is a stack of disks called platters with a magnetic coating on each surface. Tiny magnetic recording heads on the end of an arm float a millionth of a centimetre above the disk spinning at 110 km/h underneath.

Data is recorded on each disk along circular tracks, each split into smaller parts called sectors. When data is read:

- the arm moves across to be above the right track
- the required sector comes around under the head
- the surface behaving like a magnet causes a tiny current in
- the head the disk controller translates this into 1s and Os.

Because of these steps, data does not come from the disk immediately. Each step takes some time. The time for step 1 is called the seek time; for step 2 it is called the **latency**.

OPTICAL SECONDARY STORAGE - DVDS

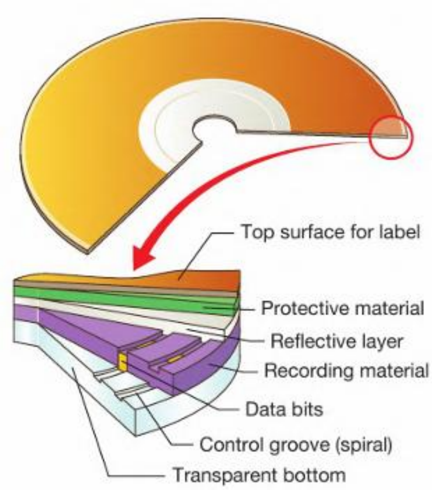
A CD, DVD or Blu-RayTM disk is made of several layers as shown in Figure 4.13. Data is written along a single track that moves out from the centre of the disk in a spiral.

If a CD is already written when it is manufactured, it has pits in the recording layer that are less reflective than the flat parts, called lands.

When reading, the laser reflects differently off the surface and this is detected by a light sensor.
The spiral track is longer on the outside so when reading data from the outside edge the disk has to reduce its rotation speed so that the data passes under the laser at a constant speed.

When data is read:

1. The disk spins in the drive to ensure all data can be read.
2. The tracking mechanism moves the laser into the correct position over the disk.
3. The laser shines on to the disk and is reflected back on to a light sensor.
4. Signals from the sensor are translated into 1s and Os.



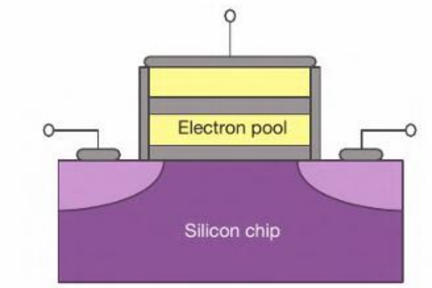
▲ **Figure 4.13** The structure of a DVD

ELECTRICAL SECONDARY STORAGE - FLASH DRIVES

Solid-state storage uses chips (called NAND flash) made of special kinds of transistors that can trap electrons in a 'pool'. Electrons in a pool represent the Os of the data, while empty pools represent 1s.

When data is read from the chip:

1. Control signals identify which bit is to be read out and apply a small voltage.
2. If the electron pool is empty the transistor turns on and a 1 is read out.
3. If the electron pool is full the transistor doesn't turn on and a 0 is read out.
4. The control signals are changed to read other bits.
5. When data is written to the chip, control signals identify which bit is to be written and apply a higher voltage.
6. This pulls electrons into the pools of those transistors, recording the 1s and Os.



▲ **Figure 4.14** Flash drive

Erasing data also requires higher voltages to remove electrons from the pools. Because of this, both erasing and writing cause the transistor to break down slowly. This means that flash drives can only be rewritten around 1 million times before eventually failing.

SUBJECT VOCABULARY

cloud storage secondary storage, often belonging to a third party, that is accessed via a network, usually the Internet, and so is not in the same physical place as the machine's RAM/ROM. Files stored 'in the cloud' can be accessed from anywhere via an Internet connection

virtualisation any process that hides the true physical nature of a computing resource, making it look different, usually to simplify the way it is accessed

Cloud storage

Sometimes it can be useful to have your secondary storage in a different place from your computer. For example, you might want to have a back-up copy of your data in a secure location or share your data with users of other computers; or you might want to be able to access your data when you are not sitting at your usual computer.

Sometimes storage is accessed via the Internet using Internet services such as Dropbox® or Google Drive. In this situation it is usually called **cloud storage**. This is an example of **virtualisation**.

Storing data on servers accessed via the Internet has the following advantages:

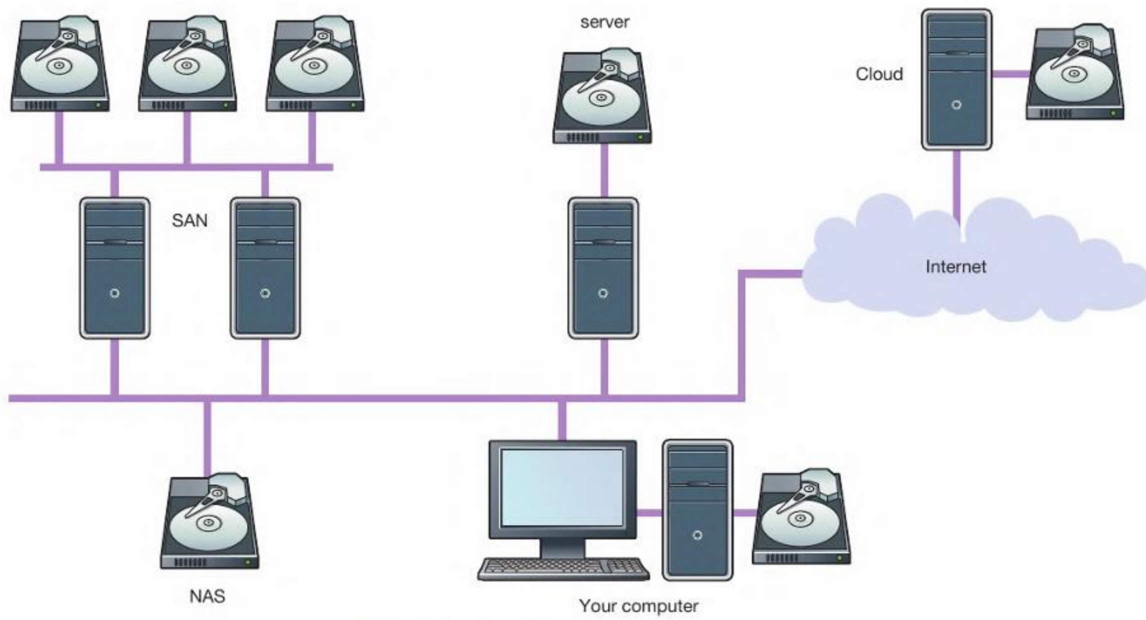
■You can access the data from anywhere on many devices using a web browser.

■The data is securely backed up by the company providing the storage service.

■You don't need to transfer your data if you get a new computer.

Some people will be concerned about security: you might, for example, have read about high-profile cases of private photographs being accessed and released on the Internet.

Other people might be uncomfortable with not being able to have their data 'in their hand.



▲ **Figure 4.15** Cloud storage diagram

Embedded system

Embedded systems are designed for specific tasks and are part of larger systems, such as washing machines.

In recent years, wireless network technology, the Internet and embedded systems have come together (converged). Many devices can now communicate with each other and be accessed and controlled remotely via the Internet. This has become known as the **'Internet of things'**.

SUBJECT VOCABULARY

Internet of things (IoT) the interconnection of digital devices embedded in everyday objects

Chapter 18 – Logic

TRUTH TABLES

You learned in Unit 1 (pages 12–13) about selection – determining which parts of a program run, depending on certain conditions – usually expressed using `IF` statements.

```
IF health <= 0 THEN
    SET game_over TO true
END IF
```

In general, this kind of statement has the form:

```
IF CONDITION THEN COMMANDS
```

GENERAL VOCABULARY

condition something that must happen before something else can happen

command an order to a computer

When this code is translated the logic statement or **condition** is calculated. If the result is TRUE then the **commands** will run.

Here is a more complicated condition.

```
IF health <= 0 AND lives = 0 THEN
    SET game_over TO true
END IF
```

▶ **Table 4.1** Truth table for the AND operator

HEALTH <= 0	LIVES = 0	GAME OVER?
No	No	No
No	Yes	No
Yes	No	No
Yes	Yes	Yes

SUBJECT VOCABULARY

truth table a table showing all possible combinations of the inputs and outputs of an operator

Boolean something that can take only the values True or False; named after English mathematician George Boole

This kind of table is called a **truth table**. It is the truth table for the AND operator. The outcome is only true if both of the conditions are true. (Usually you will see a 'Yes' written as a '1' and a 'No' written as a '0'.)

Perhaps the game doesn't have lives and we only want to end the game if you run out of 'health' or you pass the winning score of 1 million.

```
IF health <= 0 OR score > 1000000 THEN
   SET game_over TO true
END IF
```

Here are the possibilities – this time it is the truth table for the OR operator (Table 4.2).

▶ **Table 4.2** Truth table for the OR operator

HEALTH <= 0	SCORE > 1 000 000	GAME OVER?
0	0	0
0	1	1
1	0	1
1	1	1

In this case the outcome is true if either of the conditions is true.

INPUT	OUTPUT
0	1
1	0

▲ **Table 4.3** Truth table for the NOT operator

There is one more logical operator (or **Boolean** operator) called NOT (Table 4.3). This operator just swaps 1s and 0s. Notice that, unlike AND and OR, NOT has only one input.

We could use this in our 'game over' example as follows.

```
IF (NOT god_mode AND health <= 0) OR score > 1000000 THEN
   SET game_over TO true
END IF
```

We can list all the possible combinations in a truth table as in Table 4.4. (Like in maths, brackets are evaluated first.)

SUBJECT VOCABULARY

logic circuit an electronic circuit that has inputs and outputs that follow one of the Boolean operators

bug an error or flaw in a computer program

GOD_MODE	NOT GOD_MODE	HEALTH <= 0	NOT GOD_MODE AND HEALTH <= 0	SCORE > 1000000	GAME OVER?
0	1	0	0	0	0
0	1	0	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	0	1	0	1	1

▲ **Table 4.4** Truth table showing all the possible combinations

The three logic operations `AND`, `OR` and `NOT` are very easy to build as electronic circuits known as **logic circuits**.

Writing logic statements

WRITING LOGIC STATEMENTS

When you write programs, you will often have to work out the correct logic statements for a given situation so that the right code runs. Mistakes in the logic (logic errors) are a frequent cause of **bugs**. So, it is important to check that the logic is right and that the code matches the logic you have worked out.

SUBJECT VOCABULARY

operator precedence the order in which you apply the operators (including logical operators) in a mathematical equation

Chapter 19 – Software

Computer systems consist of both hardware and **software**. Software is the set of programs that run on a computer system. This is **application software**. The other kind of software is there to help us use the computer and the application programs, such as providing the user interface and tools to manage the hardware. This is **system software**, and it is usually divided into **operating systems and utility software.**

SUBJECT VOCABULARY

software the set of programs run by a computer system

application software software that performs a task that would otherwise be done by hand, perhaps with pen and paper

operating system software designed for particular hardware and which manages other programs' access to the hardware

utility software software that does a useful job for the user that is not essential to the operating system and not the reason for using a computer in the first place

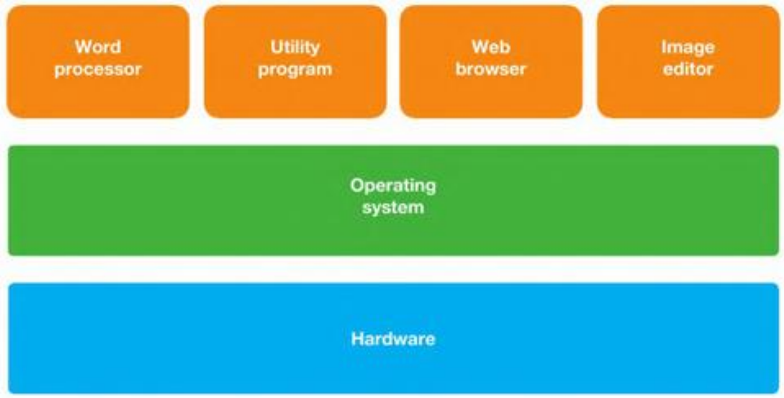
system software operating system and utility software

Operating system

The operating system (OS) is usually loaded when the computer starts up.

Why do you need this software? Unlike embedded systems, a general- purpose computer can be used to do a range of jobs. Software written for general-purpose computers needs to be able to run on a range of different hardware.

This will depend on what sort of secondary storage the file happens to be on. You might have realised this is yet another example of abstraction.
This system means applications have to be written for a particular operating system but will work on any hardware.



▶ **Figure 4.16** The operating system sits between the hardware and the programs

In general, managing files and the directory structure, and input/output, is one of the important jobs of the operating system. Two important pieces of hardware that have to be shared are the CPU and RAM.

SUBJECT VOCABULARY

application programming interface (API) code that allows two programs to communicate with each other

scheduling the algorithm that the OS uses to allow each running process to use the CPU

paging the algorithm the OS uses to move programs from RAM to disk and back again when needed once main memory is full

■In the chapter on hardware, you learned that a von Neumann system does only one thing at a time. The operating system keeps up this illusion by allowing each program to use the CPU for a short time before switching to the next program. This is called scheduling.

■It is possible to load many applications all at the same time and each uses some memory. What if you keep loading applications until RAM is full? In this case the operating system creates another illusion, by moving programs from RAM to disk and back again when needed. This is called paging, swapping or virtual memory.

They include various parts of the operating system itself such as the programs that manage the network, the scheduler and also any hardware drivers and utility programs such as anti- virus. Each task that is running is called a process. To see all the processes that are running, including the invisible ones:

■on Microsoft Windows, press CTRL-ALT-DEL and choose Task Manager
■on Linux, type ps aux at the command prompt
■on OS X, run the Activity Monitor from Applications/Utilities.

Information about how much the hardware is used by each process is displayed, including:

- the proportion of the CPU time spent on this process
- the amount of memory used by this process
- the amount of available memory (real memory and virtual memory, VM).

These processes are running one at a time, being given their share of the CPU time by the scheduler. They are all running alongside one another, but not really at the same time, so we say they are running **concurrently**.

In a multi-user system, each user has to identify themselves (with a username, fingerprint, etc.) and provide a password (or other login) to prove they are who they say they are. This is called **authentication**.

The operating system also provides and runs the **user interface**. Sometimes this is a graphical user interface (GUI) with windows, icons, menus and pointers (WIMP) like Microsoft Windows or MacOS. Sometimes it is just a text, command-line interface (CLI) like MS DOS.

> **SUBJECT VOCABULARY**
>
> **concurrent** processes that run at the same time are described as being concurrent
>
> **authentication** the process of proving to a computer system who you are (e.g. using a username and password)
>
> **user interface** the way the user interacts with the operating system

Utility software

Utility software is software that does a useful job for the user. It is:

■not essential to the operating system
■not the reason for using a computer in the first place (i.e. it is not application software).

- Utility software, sometimes called tools, can be split into three areas:
- basic tools
- file management security.

> **SUBJECT VOCABULARY**
>
> **back-up** a copy of files in another location so that they are still available if the original copy is damaged or lost. Backing up is the process of making a back-up copy

BASIC TOOLS

Basic tools include things like a simple text editor (e.g. Notepad or nano), calculator, command prompt, hex editor and software for accessibility, such as for producing large print. They are usually included as part of the operating system.

FILE MANAGEMENT TOOLS

File management tools include software to keep your data secure by making a **back-up** copy of your files in another location. Some programs can recover files that have been deleted or repair files that might be damaged or have errors.

Another useful file management tool is for converting files between different formats. This is especially useful for multimedia files where there are a large number of common formats (e.g. jpg, gif, png).

Another important file management tool is a **defragmenter** which is used to speed up access to data stored on magnetic hard disk drives. Files are not stored on disk in one large chunk. They are stored in smaller pieces called sectors or clusters and a single file's clusters can be spread across the disk.

SECURITY TOOLS

Security tools include anti-malware programs such as antivirus and anti- **spyware** software. Computer viruses and spyware can be used to capture private information, to introduce a process on your computer to do a job for the attacker, such as send out spam email, or to crash an important machine such as a web server.

Firewall software controls your connection to a network, deciding which network data can flow to and from your computer. This can help to prevent infection by viruses.

SIMULATION AND MODELLING

In Unit 1 you learned about how you can use abstraction to model the real world. By now you will be getting used to the idea of abstraction as a central part of computer science. It means hiding complexity, only exposing the key features of a situation and leaving out the details.

You can, however, write a computer program to model or simulate these situations and see what results you get. Scientists, engineers, economists and other experts can ask **'what if?'** questions using computer models.

There are two big problems with this approach:

■The model or simulation includes assumptions. It is not reality, so the answers might not be right.

■The real world is far too complicated to allow for every possible factor in your model. So, you have to use abstraction to simplify it. This makes it even less likely that you will get a completely right answer.

SUBJECT VOCABULARY

'what if?' question running a computer model with a given set of inputs to see what the model produces as an output or prediction

EXAMPLES OF COMPUTER MODELS

If you ever watch the weather forecast, you will sometimes hear the presenter talking about their computer and sometimes even about computer models. The atmospheric models used to forecast the weather are some of the most complex pieces of software that exist.

Another example of a computer model is used for earthquake prediction. Unlike the previous examples, we don't know the mathematics behind how earthquakes work. Models like these have to use information from past examples and probability to come up with best guesses as to what might happen and when. Some common techniques used in such models are **heuristics, Monte Carlo methods and neural networks**. There is a whole field of computer science devoted to finding patterns in data and using them to predict the future: this is machine learning.

SUBJECT VOCABULARY

heuristic a type of algorithm capable of finding a solution to a problem quickly and easily. This is done through **trial and error** and educated guesswork to cut corners and remove less likely alternatives. Heuristic algorithms don't always find the best solution, but they will usually find one that works

Monte Carlo method carrying out a statistical analysis of a number of random samples to get approximate solutions to a problem. The larger the number of samples used, the more accurate the result is likely to be

neural networks process information in a similar way to human brains and learn and adapt over time. This makes them useful tools for recognising faces, identifying illnesses and quality control, for example. Computers are normally not very good at these things

Chapter 20 – programming languages

Low level programming language

One other thing you learned is that the CPU can only do a few simple things. It only 'understands' a few very simple instructions. Complex things like playing games are possible because a really large number of those simple things happen very fast. Those few simple instructions that a CPU knows how to do are called the **instruction set** for that type of CPU. Each instruction is given a binary code. It is these codes that make up your programs. The binary codes representing a program are called **machine code.**

A machine code program would look something like the diagram below. You can probably see why programming in machine code would be completely impractical.

0110100100100101010101011111010101100101011010110101010

▲ Figure 4.17 An example of a machine code program

Not only would it be incredibly difficult to work out the sequence of 1s and Os you need, but also your program would only work on that kind of CPU. This is because different CPUs have different instruction sets. Instead, you need a **translator** which converts your programs into the CPU's machine code language.

Translators are also programs. Their input is the text of your program - this is called your **source code**. Their output is machine code which the CPU can run (execute).

The simplest translator is called an **assembler**. It converts assembly language to machine code. **Assembly language** is called a **low-level programming language.**

Assembly language is a bit easier to work with than machine code because each instruction is written as a short, memorable keyword called a **mnemonic.**

SUBJECT VOCABULARY

mnemonic a short, simple acronym tha represents each of the instructions in a CPU's instruction set, e.g. LDR (load register), STR (store) and CMP (compare)

emulator hardware or software that allows one type of computer system to behave like another

reboot to start a device again, e.g. turn off the computer and turn it on again

Writing programs in assembly language is challenging for three reasons.

- A very limited range of instructions is available. Every task, even the simplest, has to be built up from the smallest steps. Some older CPUS could not even multiply numbers.
- You have to manage all your data. There are no strings, integers or real numbers, just binary, so you have to decide how to represent your data. You also have to decide and manage where it is stored in memory. Debugging is very difficult.
- When the assembled program runs, any bugs usually just make the machine crash, and you have to **reboot** to try again.

High level languages

It is much more common to write software in high-level programming languages such as Python, C#, Java or Visual Basic. For these programs to run on the CPU, the source code has to be translated into machine code. This can be done all at once, and the finished machine code program saved and run later, one line at a time.

COMPILERS AND INTERPRETERS

A translator that translates the whole program in one go is called a compiler. A translator that translates and runs your program one line at a time is called an **interpreter.**

A compiler converts high-level language to machine code and saves the output as a machine code program, sometimes called **object code**.

▶ **Table 4.5** Using compilers or interpreters – a comparison

INTERPRETER	COMPILER
Every computer that will run your program needs the interpreter software installed.	The output from a compiler will run on its own on any similar computer.
Interpreters find errors when they happen and can often tell you what has gone wrong.	A compiler cannot produce any object code unless the whole program is correct – they tend to report a lot of errors initially, making it harder to debug your program.
Programs tend to run slower using an interpreter because the interpreter has to translate the source code while the program is running.	It can be easier to protect your code from being altered or copied if it has been compiled because you only give people the object code (machine code), which is hard to understand.

SUBJECT VOCABULARY

compiler a translator that converts high-level language source code into object code, often machine code. The source code is translated all at once and saved to be executed later

interpreter a translator that converts high-level language source code into object code, often machine code. The source code is translated and executed one line at a time

object code the translated source code. Often this will be machine code, but might also be an intermediate code, which has to be further translated before it can be executed

Revision questions

2023June

1).

(d) Programmers write and test code.

They use modular testing.

State what is meant by the term modular testing. (1)

2).

(c) Software is divided into two categories.

Describe one difference between system software and application software.(2)

(d) Here is an image of secondary storage.

Two files (W and Z) are stored on it.

Each file is made up of several blocks (e.g. Z1, Z2, Z3).

	W4	Z1					W2			W3	Z3		W1		Z2

Complete the image to show the state after running a defragmentation utility. (2)

(f) A series of tasks is written in different programming languages.

Complete the table to show the correct programming language translator for each task. (4)

Task	Translator
A guessing game that can be used on different computing platforms	
A screen driver for a new smartphone	
A new version of a spreadsheet program for sale next year	
Control software for an embedded system inside a new washing machine	

3).

Hardware devices execute programs to carry out a variety of tasks.

(a) A program controls a bee character in an animation.

The bee can turn to face North, East, South or West.

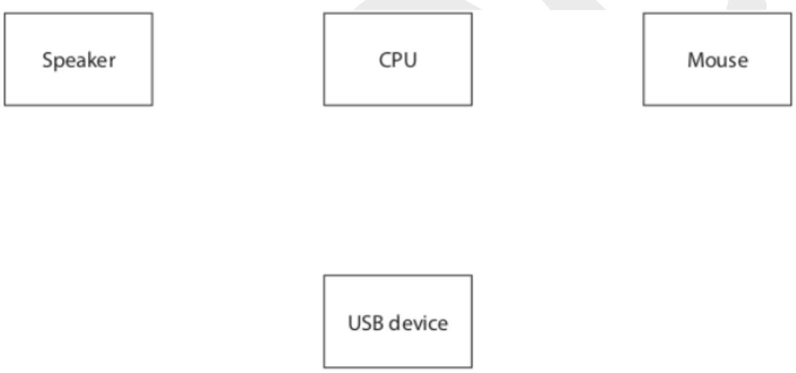
The bee can move any number of steps in the direction it is facing.

Complete the table to show one input and one output required to move the bee. (2)

Requirement	Example
Input	
Process	Calculate the path the bee will move along to its new position
Output	

(b) Computers are made up of hardware components.

(i) Complete the diagram by adding directional arrows between the components to show the flow of communication.



(ii) Identify what is stored in ROM.

- A The software firewall
- B The basic input output system
- C The operating system
- D The user interface code (1)

(iii) Cache is used as temporary storage.

One type of cache is located between main memory and the CPU.

Explain one reason cache is used in a computer. (2)

(c) A washing machine uses several different embedded systems.

One embedded system uses a switch to identify the type of wash cycle selected by the user.

Describe one other example of an embedded system found in a washing machine. (2)
(d) Programs use logic statements to control physical hardware.

(i) A window shuts when the temperature is too cool or it is a rainy night.
The values are defined as:

- A shows it is night time
- B shows it is too cool
- C shows it is raining.

Complete the truth table to show the results of each operation.
Two rows have been done for you. (3)

A	B	C	A AND C	(A AND C) OR B
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1	1	1

(ii) A warehouse has an automated alarm system.
When the alarm system is activated it will sound if:

- a movement sensor (M) is activated
- a pressure pad (P) is activated
- a key code (C) to deactivate the alarm system has not been entered.

Construct a logic statement, using AND, OR and NOT with the letters M, P and C, to show the conditions that will sound the alarm. (3)

(e) A computer with a single CPU runs several processes at the same time.
This computer is multitasking.
Describe how the operating system enables processes to share a single CPU. (2)

2022June
4).
1. Computer systems have both hardware and software components.
(a) The central processing unit (CPU) uses the fetch-decode-execute cycle.

(i) State what is meant by the term program instruction. (1)

(ii) State what is meant by the term memory address.(1)

(b) Identify the component of the CPU that provides temporary data storage.(1)

☐ A Address bus

☐ B Data bus

☐ C Control unit

☐ D Register

(c) The performance of the CPU is affected by the clock speed.

(i) Give one benefit of having a higher clock speed. (1)

(ii) Give one drawback of having a higher clock speed. (1)

(d) Identify which **one** of these describes a sequential computational model. (1)

□ A Program instruction is read one after another from external storage

□ B Program instructions are executed by multiple agents working together

□ C Program instructions are executed in parallel by different cores

□ D Program instructions are executed one after another

(e) A program can be written in a high-level or a low-level language.

(i) Give **one** reason for writing a program in a low-level language. (1)

(ii) State the purpose of an assembler. (1)

(iii) Complete the table by adding **one** tick (✓) in **each** row to match the description. (3)

Description	Compiler	Interpreter
Translates the program each time it is executed		
Produces permanent object code		
Translates line by line		
Translates the whole program before it is run		
Generates a list of errors once the complete program has been translated		

5). (d) Doctors use laptops when they visit patients in their homes.

(i) The laptops have solid state drives.
Explain one reason why a solid state drive is better than a magnetic hard drive for the laptops. (2)

(ii) Describe how data is stored on a solid state drive. (2)

(iii) The laptops have two types of memory.
Complete the table by adding one tick (√) to match each description to the type of memory used. (2)

Description	RAM	ROM
Stores the boot up sequence		
The contents are lost when the laptop is shut down		

2021Nov

6).
Computers can be general-purpose or specific-purpose machines.

(a) The central processing unit (CPU) is responsible for executing a program.
Figure 4 shows some of the hardware components of a computer system.
Complete the table by matching the component to the letters shown in Figure 4. (5)

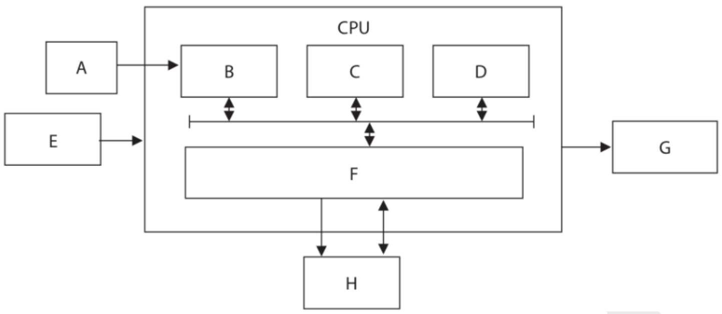


Figure 4

Component	Letter
Control unit	
Output device	
Registers	C
Main memory	
Cache	F
Input device	
Arithmetic/logic unit (ALU)	
Clock	A

(b) The CPU runs a fetch-decode-execute cycle.

(i) Some registers within the CPU are general-purpose whilst others perform a specific function.
One register is the program counter. Its function is to keep track of the next instruction to be fetched.
Describe how it does this. (2)

(ii) The control bus carries signals from the processor to other components.
Identify the signal that would be used during the fetch stage of the cycle. (1)

- A Memory delete signal
- B Memory read signal
- C Memory store signal
- D Memory write signal

(iii) Clock speed is the number of pulses the CPU's clock generates per second.
State how increasing the speed of the clock impacts on the fetch-decode-execute cycle. (1)

(c) An embedded system forms part of a larger system, device or machine.

(i) Give two features of an embedded system. (2)

(ii) There are many factors to consider when choosing a CPU for an embedded system.
For example, the clock speed of the CPU in a TV remote control can be low because it only has to be fast enough to create a low frequency output signal for the TV.

Explain one other factor that should be considered when choosing a CPU for an embedded system within a TV remote control. (2)

7).
Akiko develops software programs.
(a) Akiko can use a low-level or a high-level programming language to write code.

(i) Give one benefit of using a low-level programming language to develop a program. (1)

(ii) Give one drawback of using a low-level programming language to develop a program. (1)

(b) Akiko is writing an anti-virus program.

(i) The program will include a virus removal function.
Give two other functions that should be included. (2)

(ii) An anti-virus program is one type of anti-malware.
Give one other type of anti-malware. (1)

(c) Akiko is also working on a burglar alarm system.
The alarm must only be triggered if:

the alarm (A) is set
it is dark outside (O)
a window (W) or a door (D) has been opened.
Construct a Boolean expression, using AND, OR and NOT with the letters
A, O, W, and D to show the conditions that will trigger the alarm.(4)