Name: Index: Reg. No.: Class: Date:

Chapter 4

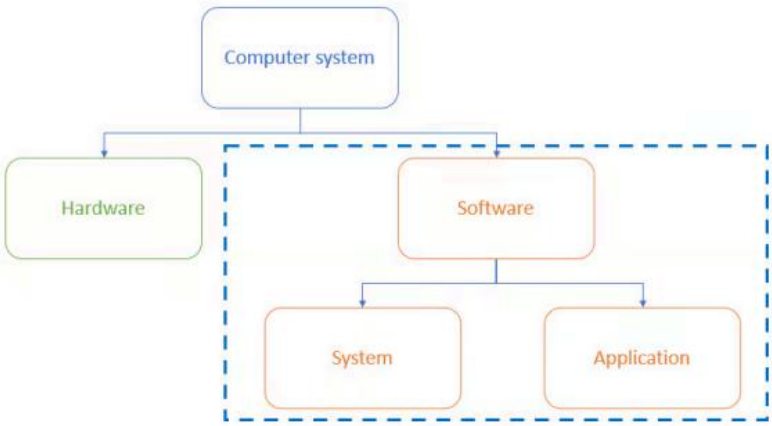
Software

What is Software?

Software is a set of **programs and instructions** that tells a computer **what to do and how to perform tasks**.

System Software and Application Software

- Software can be broken down in to two categories, **system** and **application** software



What is System Software?

- **System software** is software that **controls and manages the computer's hardware** and provides a platform for application software to run.
- Provides a **human computer interface (HCI)**
- Examples of system software include:
  - The operating system
  - Utility software

Utility Software

- Utility software is software designed to help **maintain, protect** and **efficiently run** computer system
- Designed to **perform a limited number of tasks**
- Examples of utility software and their function are:
  - **Security utilities** (anti-virus, encryption, firewall)

- **Disk Defragmentation Tools** – rearrange the parts of files on the disk drive
  - When a file is saved to the **HDD**, parts of the file might be saved in different areas of the disk → these tools try to move all the parts to the same area for quicker access
- **Data compression utilities** (makes files smaller to take up less storage space for faster transmission)
- **File backup utilities** (backs up files and software on the system)

What is Application Software?

- Application software are the software which provides the services that the user requires
- Allows a user to perform **specific tasks** using the computer's resources
- Common categories of application software include:
  - **Productivity** - get things done efficiently (word processors, spreadsheets & presentation)
  - **Communication** - stay connected (email, browser, messaging)
  - **Entertainment** - Watch movies, play games or listen to music

Examiner Tips and Tricks

Do not use **brand names** when talking about software, it is ok to talk about software categories or software types but brand names do not get awarded marks

e.g. word processors instead in Microsoft Word

The Purpose & Functionality of Operating Systems

What is an operating system?

- An operating system (**OS**) is software that **manages computer hardware** and **provides a platform for running applications**
- It provides **an interface between the user and the hardware** in a computer system
- An operating systems **main functions** can be divided in to **eight** key areas
  - **File Management**
  - **Handling Interrupts**
  - **User Interface**
  - **Peripheral Management & Device Drivers**
  - **Memory Management & Multitasking**
  - **Providing a Platform for Running Applications**
  - **Providing System Security**
  - **User Management**

1. File Management

- File management is the process of **creating**, **organising**, **renaming**, and **controlling** access to files and folders.
- The OS allows users to control who can access, modify and delete files/folders (**permissions**)
- File management gives the user the ability to:
  - **Create files/folders**
  - **Name files/folders**
  - **Rename files/folders**
  - **Copy files/folders**
  - **Move files/folders**
  - **Delete files/folders**

2. Handling Interrupts

- An **interrupt** is a signal from **hardware** or **software** to the **CPU** indicating that attention is needed
- This will cause the CPU to **temporarily stop** its current task to **handle the interrupt**
- **The OS** processes interrupts quickly **to keep the system running smoothly.**
- For example, Clicking **Cancel** during file conversion sends an interrupt, causing the OS to stop the conversion.

3. User Interface

- A user interface is **how the user interacts with the operating system**
- Examples of user interface includes **Command Line Interface (CLI)** or a **Graphical User Interface (GUI)**.

What is a command line interface (CLI)?

- A Command Line Interface (CLI) requires users to interact with the operating system using **text based commands**
- The user has to therefore learn a number of commands just to carry out basic operations

- CLIs are more commonly used by **advanced users**
- Examples of CLIs are MSDOS (Microsoft Disk Operating System) and Raspbian (for Raspberry Pi)

What is a graphical user interface (GUI)?

- A Graphical User Interface (**GUI**) requires users to interact with the computer using **visual elements** such as windows, icons, menus & pointers rather than having to type in a number of commands.
- GUIs are **optimised for mouse and touch gesture input**
- Examples of GUIs are Windows, Android and MAC OS

Advantages and disadvantages of user interfaces

Interface	Advantages	Disadvantages
Command line (**CLI**)	▪ Uses less system resources ▪ Useful for automation of tasks ▪ Commands are often faster to type than navigating menus	▪ Requires users to remember commands ▪ Typing errors are common Less intuitive than GUI
Graphical (**GUI**)	▪ Intuitive and user-friendly ▪ Requires no previous knowledge to use ▪ Information is visual, making it easier to understand	▪ Uses more system resources ▪ Can be slower to find and execute commands ▪ Can be frustrating when doing repetitive tasks

4. Peripheral Management & Device Drivers

What is peripheral management?

- Peripheral management is a process carried out by the operating system **managing the way peripherals (hardware) interact with software**
- The OS allocates system resources to peripherals to ensure **efficient operation**
- Peripheral management makes **plug-and-play** (PnP) functionality possible, **automatically detecting and configuring new peripherals** without the need for manually installing device drivers or power cycling the system

What is device driver?

- A device driver is **a piece of software used to control a piece of hardware**
- Peripherals **require device drivers** in order to be used by the operating system
- The OS has **generic device drivers built in** which makes **basic compatibility** possible and enables **plug-and-play** (PnP)
- In order for hardware to be used to its **maximum capacity**, often **a separate device driver must be downloaded** from the manufacturer
- Device drivers are **OS specific** and are **regularly updated**

5. Memory Management & Multitasking

What is memory management?

- Makes sure that two processes don't try to access the same memory location
- Managing the movement of data to and from RAM
- Checks that processes have enough memory located to them
- Manage the transfer of pages between virtual memory and RAM
- Memory management makes **multitasking** possible

What is multitasking?

- Multitasking allows a computer to execute multiple tasks simultaneously by quickly switching between them.
- The operating system manages multitasking by allocating CPU time to each task in a way that users feel tasks are running simultaneously.
- The CPU can only execute one instruction at a time, it can execute billions of them in one second
- This makes it appear that multiple programs are running at the same time
- **Example**: Browsing the web, playing music, and downloading a file simultaneously on the same computer.

6. Providing a Platform for Running Applications

- Operating systems provide a **platform** on which **application software can run**, this is mainly by allowing software **access to system resources**
- For example, if a computer game has intensive graphics and online play, the operating system will grant it access to the GPU and the network card

7. Providing System Security

- The operating system protects the computer, its users, and data from unauthorised access.
- **Creates and deletes user accounts** – manages who can use the computer.
- **Controls access permissions** – decides what each user is allowed to do.
- **Uses passwords** – prevents unauthorised users from accessing accounts.
- **Installing security updates** to protect against threats

8. User Management

- User management is a process carried out by the operating system enabling different users to log onto a computer
- The OS is able to maintain settings for individual users, such as desktop backgrounds, icons and colour schemes
- A system administrator is able to allocate different **access rights** for different users on a network

Worked Example

Ella uses her computer to create artwork for a magazine

Ella makes use of system software.

One type of system software is the operating system.

Identify and describe **two functions** of an operating system [6]

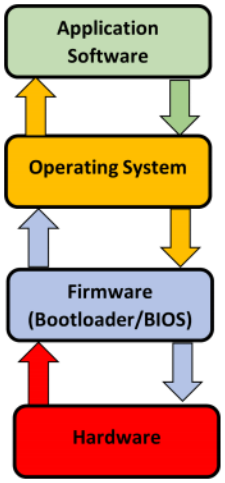
How to answer this question

- Break down the 6 marks, 1 mark each for identifying a function of the operating system. For each function you need to make 2 points about how they work

Answers

- Memory management
  - Allocates memory to programs currently in use
  - Gets data from RAM
  - Stores data in RAM
- File management
  - Creating/editing/renaming files
  - Creating/editing/renaming folders
  - Movement of files/folders

How does application software, the operating system and hardware communicate?



- When the computer is switched on, **firmware (BIOS/UEFI)** checks and initialises the hardware and loads the operating system.
- The **operating system runs in RAM** and manages the computer's hardware resources.
- When an application needs to use hardware, **it requests the required services from the operating system.**
- The operating system **communicates with and controls the hardware** to carry out the task requested by the application.
- The **hardware performs the task** and sends the result back through the operating system to the application.
- This process **continues repeatedly while the application is running**.

What is firmware?

- **Firmware** is software stored in **non-volatile memory** inside a hardware device. It helps **control and initialise the hardware.**
- On a computer, the **BIOS/UEFI** is firmware that runs when the computer starts **and initialises the hardware.**
- Firmware provides a **link between hardware and software**, allowing software to interact with the hardware.
- **Examples:** BIOS/UEFI, printer firmware, keyboard firmware, router firmware.

Interrupts

What is an interrupt?

- An interrupt is a signal sent from a device/software to the CPU, to temporarily stop what it is currently doing so that it can service the interrupt.

Types of interrupts

An interrupt can be generated by hardware and software:

- **Hardware** - this is caused by a hardware device
  e.g. pressing a key on the keyboard, moving the mouse
- **Software** - this occurs when an application stops or requests services from the OS
  e.g. division by zero, two processes trying to access the same memory location

Operations of interrupts

- Hardware/Software generates the interrupt
- Interrupt is given a **priority**
- Interrupt is sent to **CPU** and placed in a **queue**
- CPU stops current task to service the interrupt using an **interrupt service routine**
- If interrupt is highest priority the interrupt is processed
- Once the interrupt has been serviced, the **previous instructions** are retrieved and the process continues

Advantages of interrupts

- The system becomes more **responsive** to real-time events (like user inputs or hardware failures).
- It allows **multitasking**, by switching between different tasks quickly.
- Critical tasks are handled **immediately**, improving performance and safety in systems.

What happens without interrupts

- The computer would be less efficient.
- The CPU might take longer to respond to events.
- Multitasking would be less efficient.

Worked Example

Describe the purpose of an interrupt in a computer system

Answers

Four from:

- Used to attend to certain tasks/issues
- Used to make sure that vital tasks are dealt with immediately
- The interrupt/signal tells the CPU/processor (that its attention is required)
- The interrupt will cause the OS/current process to pause
- It enables multi-tasking to be carried out on a computer

Types of Programming Language

What is a programming language?

- A programming language is a language used to write programs that tell a computer what to do.
- Generations of programming languages can be split in to two categories:
  - **Low-level**
  - **High-level**

1. Low-Level Languages

- A low-level language is a programming language that are close to the **language processed by computers**
- Low-level languages allow **direct control over hardware** components such as **memory** and **registers**
- Low-level languages can refer to **machine code** or **assembly language/code**

1. Machine Code

- Machine code is the **binary code**
- Instructions are **directly executable by the processor**

2. Assembly Language/Code

- Assembly language is a form of **low-level language** that uses **mnemonics**
- **Needs to be translated** into machine code for the computer to be able to execute it
- **Assembler** is the translator used in assembly language
- **Example of mnemonics:** MOV AX, 5

Advantages and disadvantages of low-level languages

Advantages	Disadvantages
Can directly manipulate the hardware	Difficult to write and understand
Faster execution due to no requirement of compiler/interpreter	Machine dependent
Can use specialized hardware	More prone to errors
More memory efficient	Harder to debug

2. High-Level Languages

- A high-level programming language are languages which uses **English-like statements**
- This language **needs to be translated** into machine code
- Examples of high-level languages include:
  - Python
  - Java
  - C+

Advantages and disadvantages of high-level languages

Advantages	Disadvantages
Easier to read and write	It is not directly manipulate the hardware
Easier to debug	Needs to be translated to machine code before running
Portable so can be used on many computers	Slower execution time compared to low level languages
Greater range of languages are available	Take up more memory

Example of computer code

Computer code	High-level	Assembly	Machine code
10110111 00110110 11100110			✓
FOR X = 1 to 10 PRINT x NEXT X	✓		
INP X STA X LDA Y		✓	

Translators, Compilers & Interpreters

What is a translator?

- A translator is a program that **translates** program **source code** into **machine code** so that it can executed directly by a processor
- **Low-level languages** such as **assembly code** are translated using an **assembler**
- **High-level languages** such as **Python** are translated using a **compiler** or **interpreter**

What is a compiler?

- A compiler translates **high-level languages** into **machine code all in one go**
- A compiler creates an **executable file**
- Produce error reports for the **whole code**
- Once a program is compiled, the executable file can be used again to perform the same task **without re-compilation**
- If any changes made after compiling, it **will need re-compiling again**
- Compilers are generally used when **a program is finished**

Advantages and disadvantages of compiler

Advantages	Disadvantages
Program take less time to execute	Compilation can take time, especially for large programs
The compiled program can be executed without the compiler	Errors need to be corrected before the program can run
The source code is not included in the executable file, making it harder to change the program	Changes mean it must be recompiled again
Executable file can be distributed to others	Compilation requires additional memory to process the program

What is an interpreter?

- An interpreter translates high-level languages into machine code **one line at a time**
- Each line is executed after translation
- **Stop execution** when any errors are found
- A program needs to **be interpreted again each time** it is run
- Interpreters are generally used when a program **is being written** in the development stage

Advantages and disadvantages of interpreter

Advantages	Disadvantages
No separate executable file needed	Slower execution as program is translated each time it runs
Easier and quicker to debug	Interpreter is required to run the program
Require less RAM to process the code	Source code is visible and can be accessed by anyone, making it less secure than compiled code.

Worked Example

A computer program is written in a high-level programming language.

(a) State why the computer needs to translate the code before it is executed. **[1]**

(b) Either a compiler or an interpreter can translate the code. Describe two differences between how a compiler and an interpreter would translate the code. **[2]**

How to answer this question

- (a) what type of language does a computer understand?
- (b) the keyword is 'how'

Answer

a)

- To convert it to binary/machine code
- The processor can only understand machine code

b)

- Compiler translates all the code in one go whereas an interpreter translates one line at a time
- Compiler creates an executable whereas an interpreter executes one line at a time
- Compiler reports an error at the end whereas an interpreter stops when it finds an error

Integrated Development Environment (IDE)

What is an IDE?

- An Integrated Development Environment (IDE) is **software** designed to make **writing high-level languages** more **efficient**
- IDEs include tools and facilities to make the **process of creating/maintaining code easier**, such as:
  - **Editor**
  - **Error diagnostics**
  - **Run-time environment**
  - **Translators**

Editor

```
main.py > ...
1  # My first program
2  name = input("Enter your name: ")
3  print("Hello, " + name + "!")
4
```

- An editor gives users an environment to write, edit and maintain high-level code

Error-diagnostics

```
main.py                                   Format
1  # My first program
2  name == input("Enter your name: ")
3  print("Hello, " + name + "!")
4
```

```
Packager  Resolving dependencies...
Run
Traceback (most recent call last):
  File "/home/runner/temp-1/main.py", line 2, in <module>
    name == input("Enter your name: ")
NameError: name 'name' is not defined
```

- Identify errors as the program is being typed and provide error messages

Run-time environment

```
main.py > ...                             Format
1  # My first program
2  name = input("Enter your name: ")
3  print("Hello, " + name + "!")
4
```

```
Run
Enter your name: Save My Exams
Hello, Save My Exams!
```

- Gives users the ability **to run** and see the **corresponding output** of a high-level language

Translators

- Provide compiler or interpreter which enables the program to be executed

Auto-corrections

- Automatically fixes certain common mistakes while you type.

Auto-completion

- It suggests or completes code as you type.

Pretty print

- Given different colors to the keywords and arrange the codes in meaningful way.