IGCSE Computer Science CIE

YOUR NOTES

4. Software

CONTENTS

YOUR NOTES

4.1 Types of Software and Interrupts

System Software & Application Software

System Software & Application Software

- Systems software provides the services that the computer requires, including operating system and utility software
- E.g. allowing instructions to be processed by the CPU to allow word processing software to process. Without systems software the system would be useless
- Systems Software is made up of two core elements: the **operating system** and **utility software**

Operating System

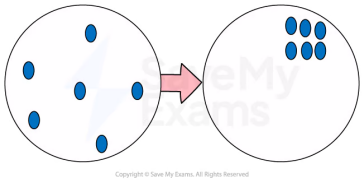
- This is a program designed to run other programs on a computer. It is considered the backbone of a computer, managing both **software** and **hardware** resources
- Operating systems are responsible for everything from the control and allocation of memory to recognising input from external devices and transmitting output to computer displays. They also manage files on computer hard drives and control peripherals, like printers and scanners.
- Examples of Operating System: Windows, MAC, Linux

Utility Software

- Utility programs perform specific tasks related to computer functions, resources, files and security. They help to configure the system, analyse how it is working and optimise it to improve its efficiency. Some of these utilities include:

- **Security utilities:**
  - **Anti-Virus** - Scans the computer system and ensures that files are quarantined so that they can be removed by the user
  - **Encryption** - uses an **algorithm** to scramble **(encrypt)** a file according to the key which is used to make the file unreadable; the key is needed to **decrypt** the file back to its original form so it can be read
  - **Firewall –** monitors network traffic and blocks unauthorised access. Criteria is set and if the data meets the criteria it is accepted otherwise it is rejected
- **Disk organisation utilities:**
  - **System Clean-Up Tools** – to search for and remove files no longer needed, to reduce space and speed up access to the system
  - **Disk Defragmentation Tools** - it is used to rearrange the parts of files on the disk drive: when a file is saved to the disk, 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

YOUR NOTES



- **Data compression utilities**
  - **File Compression Software –** to make files smaller so that they take up less storage space and can be transmitted to other users more easily
- **File backup utilities**
  - **Full Backup –** stores all files and software on the system
  - **Incremental Backup –** only backs up the files and software that have been added since the last backup

Application Software

- This is designed to carry out a specific task the user would need for completing a variety of tasks. These are the applications that the operating system would process and allow end users to complete their vital day to day tasks. Examples are:
- **Word Processing Software**
  - This allows users to create written documents
  - e.g. letters, reports etc
- **Spreadsheet Software**
  - This allows users to perform numerical calculations and function to create automation for all mathematical elements
  - e.g. budgeting, accounting, stock counts etc
- **Presentation Software**
  - This allows users to create interactive and multimedia presentations to show to an audience
  - e.g. sales pitches, launching of products etc
- **Multimedia Software**
  - This is used to create multimedia and interactive content for a number of purposes, it explores text, audio, images, video, sound and animations
  - e.g. image manipulation, editing a video etc
- **Web Browsers**
  - This allows the user to access the internet and use the world wide web

Exam Tip

- Make sure you use the names given above and not brand names e.g. Microsoft as this will cost you marks

Distribution Methods

YOUR NOTES

Freeware software is where:

- The user is **not allowed to access** the source code so, they **cannot** tailor the software to their needs or fix any bugs in it
- The software is still **covered by copyright** and the user must get the owner's permission to do anything beyond using it

Free software is where:

- The user **can access** the source code so, they **can** tailor the software to their needs and fix any bugs in it
- The source code could be studied for educational purposes
- The user can redistribute the software but this must be done under the same terms as the original software

Shareware is software which:

- Gives a trial version of the software for a limited time with limited features free of charge
- Requires the user has to pay fee if the full version is needed
- Is protected by copyright
- Is a type of software licence

There are many ethical considerations when distributing software:

- Accessibility of software
- Age appropriation
- Copyright
- Distribution of malware
- Environmental impact of distribution media e.g. CDs
- Following guidelines of professional bodies e.g. ACM/IEEE/BCS
- Intellectual property theft
- Offensive materials
- Plagiarism
- Privacy of data
- Security of software

Exam Tip

- Make sure you apply your answer to the context given in the question

YOUR NOTES

The Operating System

What does the Operating System do?

- **Managing Files:**
  - The file manager controls all of the different **files on the system**, e.g. text files, graphic files, and program files. It controls **file permissions** such as the user's ability to see or open a file, write a file or delete a file.It helps to **organise and control files** so that they are as easy to use as possible for the user. It can help to protect the user from accidental mistakes too

- **Handling Interrupts:**
  - An interrupt is a signal from a device or software to the processor. This will **stop the processor temporarily** from fulfilling this request immediately, some examples could be:
  - Software errors - e.g. files not found, or software not responding
  - The user initiating Ctrl Alt Delete to lock, log off etc
  - Files stop copying as the name of the file is already in the folder
  - Once this interrupt is received by the **CPU** it either carries on or completes the action desired
  - Whenever an interrupt is initiated the status of this task is saved to the **interrupt service routine.**
  - Once the interruption has been completed the system continues back to normal before the interruption even happened

- **Providing an Interface:**
  - Users must interact with the operating system through a **user interface**. The user interface is a system which converts what the user inputs to a form that the computer can understand and vice versa
  - Many computer or database operating systems use complex programming languages which are not easy to use
  - A user interface is created to allow easier control of the operating system by the system user. A good interface should be easy to use
  - e.g.: consistent menu structures; consistent operations from actions like clicking the right mouse button

- **Managing Peripherals and Drivers:**
  - The overall intention of this is to handle all the devices that are connected to the computer system. This includes input devices such as a keyboard and mouse, it also includes output devices such as a monitor and printer
  - It communicates with the devices through software called **drivers**. These translate the instructions by the device manager into one the devices can understand Peripherals like a mouse, keyboard and printer all need drivers so that they can communicate with the software

YOUR NOTES ⬇

- **Managing Memory:**
  - Memory management is in charge of the **RAM.** Programs use **RAM** throughout their operation. Some programs will be large and complex and use the **RAM** extensively whereas some are very small programs and won't use it as much
  - Memory management checks all requests from programs are valid and allocates accordingly. It will deallocate space and swap out data to virtual memory. It will ensure overall that different programs can be open at the same time

- **Manage Multitasking:**
  - Multitasking allows for software tasks to be completed at the same time to ensure multiple elements can be completed immediately, it uses a system called **time slicing** which splits different tasks into small segments
  - They can all be run one after the other, giving the element of multiple tasks being completed at the same time, instead of waiting for one operation to complete before moving on to the next task

- **Providing a Platform for Running Applications:**
  - Application programs and the hardware will communicate through a system within the operating system called an **application programs interface(API)**
  - This API is a library interface which will share data between software to allow elements to process
  - If any application is installed on the system, the Operating System will manage this process, it will allocate memory space and will control the application's data or devices, user access will also be managed

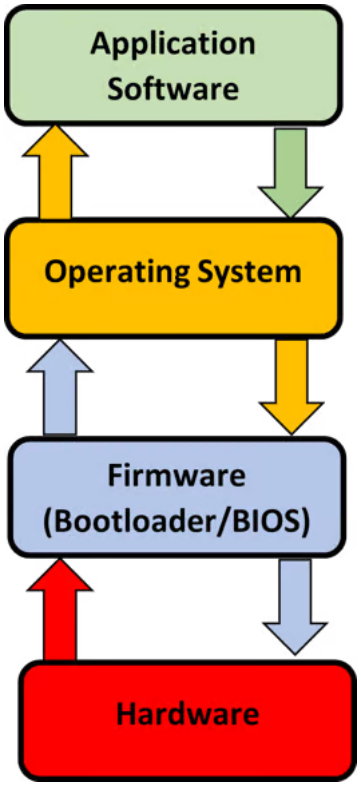
- **Providing a System Security:**
  - System security is just about how the operating system can stay protected and ensure that elements are monitored and actioned to ensure the system is secure. Some areas of security are:
    - Creating/Deleting users for the system
    - Providing access level rights, e.g. administrator rights over installing, modifying a system or accessing files or folders. This is compared to standard rights of just accessing and using software/files on the system.
    - Auditing - this is keeping a log of file edits, deleted files, creation of files etc
    - Protecting from threats, viruses, worms, malware or remote hacker attacks
    - Security updates to fix patches in the operating system

- **Managing User Accounts:**
  - Each user is provided with an account for access to the system. They will be provided with their username and then will need to create a password based on rules set out initially
  - Each account will then be granted different levels of access, dependent on needs and level of security. This will also monitor login activity and even log users out if they have been inactive for a while

Hardware, Firmware & the OS

YOUR NOTES

- Application Software must talk to the operating system, this will then allow it to interact with the hardware
- The hardware will then process and send the information to the operating system which then directly talks back to applications software in a continuous loop



- **Firmware**
  - This process is in between the Operating System and Hardware
  - When a computer initially first loads up, it has to explore the **ROM** for its initial boot-up instructions These are contained in a **Bootstrap loader**
  - The initial process is handled by the **basic input/output system (BIOS)** which is known as **firmware**. This will provide some low-level control for all devices
  - Once complete these are sent to **RAM** to be processed by the operating system
  - Overall this creates an extra layer which is to ensure that initially the hardware devices e.g. keyboard are available and can be communicated directly with the operating system
- **E.g. –** If you were to type on a word processing document, this would talk with the operating system initially to request the key presses. It would send it to the firmware to check whether the keyboard is available. Once the connection is established with the hardware it communicates directly with the keyboard back through firmware, the key instructions are passed up to the operating system and finally displayed on the word processing software

Exam Tip

- Ensure you have the understanding of how the four areas work together. An exam question would focus on how these four key areas would communicate.

YOUR NOTES

YOUR NOTES

Interrupts

- We know that computers use the **fetch-decode execute cycle** within the CPU to run instructions over and over. However, while this is occurring other devices may need to signal to the **CPU** to tell it to stop temporarily (**interrupt**) so that it can do the dedicated specific task
- E.g. a user has initiated ctrl alt delete to run task manager or a user wants a document printing
- **Interrupts** will need to ensure that the **CPU** can stop **executing** its current program to run code for the overall **interruption**. **Interrupts** need to be added to an area called the **interrupt service routine**
- **Two types of interrupt:**
    - **Hardware Interrupt –** this is caused by a hardware device such as a hardware failure e.g.
        - pressing a key on the keyboard
        - moving the mouse
    - **Software Interrupt –** this occurs when an application stops or requests services from the OS e.g.
        - a program is not responding
        - division by zero
        - two processes trying to access the same memory location
- However, the CPU currently contains registers that are holding data currently being handled: These include:
    - **Program Counter –** The location of the next instruction which will need to be fetched
    - **Current Instruction Register** - holds the current instruction being executed
    - **Memory Address Register –** stores the location of where the data is being stored in **RAM** or where in RAM the data will be sent
    - **Memory Data Register** - stores the actual data from the location in the **RAM** or that will be sent to RAM
- The **interrupt service routine** is simply added to a particular area where a certain set of instructions are sent that will need to be fetched, decoded and executed to complete the commands of the **interrupt**
- As a result, it's clear that the current **registers** will need to be changed at this point to accommodate the **interrupt**
- The **interrupt** will be **executed** instead of the original instructions
- When the **interrupt** is received the current values that were held in the registers are copied back to the **RAM** in an area known as a **stack**
- These values are pushed onto the **stack** and are added to the top of the **stack frame**, which will save them for later retrieval when the **interrupt** is complete
- There is a possibility that an **interrupt** can also be interrupted which is known as a division by zero. Due to the system in **RAM** with **stack frames**, the current **interrupt** would be moved to the bottom of the **stack frame** to complete the main **interrupt** initially

YOUR NOTES

Exam Tip

- Focus on the interrupt service routine and its importance with sending the instructions to a stack frame to process the interrupted instruction, the previous instructions can be accessed at the top of the stack frame to continue processing afterwards.

There are common **interrupt priorities** for different categories, these are:

- **Hardware**
  - Power supply may have failed
  - Power button may have been pressed
- **User**
  - Moving the mouse
  - Clicking an icon to open a new program
  - Keyboard presses e.g. ctrl, alt, delete
- **Software**
  - Illegal instruction encountered
  - Overflow
  - Login request
  - Crashing
- **Timer**
  - Data logging programs which reads sensors continuously
  - Screen recording applications
- **Input/output devices**
  - Signaling of data transfer been completed
  - Printer ink supply notifications
  - Input devices not responding

YOUR NOTES

High & Low Level Languages

Low Level Languages

- **Low Level Languages** are languages that sit close to a computer's **instruction set.** These are basic instructions that the CPU will understand
- For instance an **assembly** language which allows programmers to focus on programming simple commands, which in turn is converted into **machine** code This element is needed for the core hardware to be able to work with the software
- These languages are written for specific processors to ensure they embed the correct machine architecture
- **Assembly Language** - The code is written using **mnemonics**, abbreviated text commands such as LDA (Load), STO(Store) Using this language programmers can write human-readable programs that correspond almost exactly to machine code.
- **Machine code –** is at the hardware level and is written in **binary (1's and 0's)**

Advantages	Disadvantages
It gives programmers complete control over the system components so it can control hardware components.	Difficult to write and understand
Efficient code can be written for the processor so it will occupy less memory and execute faster	Machine dependent and cannot be added to different specification machines
They provide direct manipulation of hardware which means it will be more efficient	More prone to errors
Communicates directly with hardware	Knowledge of computer architecture is key to program effectively

High Level Languages

- **High Level Languages** are programming languages which use English-like statements which allow users to program with easy to use code, allow for clear debugging and once programs are created they become easier to maintain
- High level languages were needed more due to the development of processor speeds and memory capacity increasing

- Examples of these languages are Python, C#, Java etc

YOUR NOTES

Advantages	Disadvantages
It is easier to read and write and the programmer is less likely to make mistakes	The user is not able to directly manipulate the hardware
It is easier to debug so it will save time	Needs to be translated to machine code before running
The code is portable so can be used on any computer	The program may be less efficient
One line of code can perform multiple commands	

Exam Tip

- You will be asked about the advantages and disadvantages of high or low level languages or you will be asked to compare and contrast between the two types of language

YOUR NOTES

Assembly Language

- The first languages were actually direct machine language, where programmers had to program it with direct binary numbers of 1's and 0's. It was quite clear to see that this method was incredibly difficult to program, which allowed the introduction of Assembly languages

- Programmers who use assembly language do so for the following reasons:
  - Need to make use of specific hardware or parts of the hardware
  - To complete specific machine dependent instructions
  - To ensure that too much space is not taken up in RAM
  - To ensure code can completed much faster
- Assembly languages allow programmers to program with **mnemonics.** e.g.
  - LDA Load - this will ensure a value is added to the **accumulator**
  - ADD Addition - this will add the value input or loaded from memory to the value in the **accumulator**
  - STO, Store - stores the value in the **accumulator** in RAM

- This is used rather than binary code which allowed continuation of working directly with the hardware but removed an element of complexity
- A **mnemonic** is received by the computer and it is looked up within a specific table
- An **assembler** is needed here to check the word so that it can be converted into machine code
- If a match from the word is found e.g. STO the word is replaced with the relevant binary code to match that sequence

Exam Tip

- A question will focus more directly on how the assembler converts to work with the hardware. You must focus on mnemonics being converted from a table and into the corresponding binary code

Translators

YOUR NOTES

Compilers & Interpreters

- Programmers will write program **source code** using high-level languages e.g. Python, Java, C# etc. As programmers, we can understand **source code** as it is descriptive, easy to read, maintain and debug. However this is not good for the hardware as it needs to be converted into binary to allow the hardware to understand and execute it, this is known as **machine code.** For this to work it needs to pass through a **translator** first. There are two types of translators - a compiler and an interpreter

Compiler

- This method will translate a program into machine code. Compilers convert the source code in one go into an executable file ready for distribution. This method is used mainly when a program is finished with no syntax or logical errors
- Compiling may take time to be processed, however, this can be used over without needing to be recompiled every time, bearing in mind that the program contains no errors
- Error reports are produced after a program has been translated. Common errors in code will allow the computer to crash and not respond, it's important to be aware that if there are errors the source code must be changed to compile again

Interpreter

- This is the method that directly sends the source code to the machine code. This will translate each line of code individually, if an error occurs the program will stop and an error message will occur. Once the error message is fixed, the program can carry on running from where the error occurred

Exam Tip

- You will need to have a good understanding of the difference between a compiler and an interpreter. You will need to focus on features of compilers and interpreters or focusing on an overall comparison between the two
- Although both translators find errors in code, they do not debug the errors - this is done by the programmer

Advantages & Disadvantages

YOUR NOTES

Compiler

Advantages	Disadvantages
Run quickly as the program as the source code has been stored to be translated	Due to all code being compiled at the same time there must be enough memory space to handle this, if not and virtual memory is used it can be much slower
Compilers optimise the code, this code will run quicker and take up less memory space	If there are errors in the code the compiler will not identify directly where the error lies, making it difficult to debug
Original source code will not be seen, which is ideal for programmers to stop work being copied	It is designed solely for one specific processor
	If the program is changed it must be recompiled

Interpreter

Advantages	Disadvantages
Program will always run, it will just stop when it finds a specific **syntax error** in the code	Each line of code has to be interpreted separately by the CPU, which can lead to slower execution
It is easier to debug and understand where particular code has gone wrong	Every time the program is run it has to be translated, due to no instructions being stored
Interpreters do not store instructions and are not stored for later use, this means they require less RAM to process the code	They cannot optimise code, it is translated and executed as it is

YOUR NOTES

IDE

An **integrated development environment (IDE)** is software that consolidates basic tools required to write and test software to make a programmer's journey effective and useful, this will ensure they have key features to improve programming code and ensure it does as instructed. Some features are:

- **Basic code formatting –** changing the font, size of the font, making text bold etc
- **Coloured keywords in source code –** e.g. Python code print, input etc turn purple, if turns orange. This makes it easy to see keywords
- **Code Editing** - this will allow users to write and manipulate source code, it includes features such as auto-completion and auto-correction of code, bracket matching, syntax checks etc
- **Commenting code –** this allows sections of code to be commented out easily to stop it from being run or as comments on what the program is doing
- **Identifying errors –** highlight particular areas of code or provide direct error messages where the error may have appeared e.g. indentation errors etc
- **Run-Time environment** - to allow the program to run and see its corresponding output
- **Debugger –** this will identify and remedy errors within the source code. This can provide a step through command also which provides step by step instructions and shows what is happening to the code line by line. This method is amazing for catching **logical errors**
- **Libraries –** extra modules that are not included in the main programming language e.g. math in Python for extra mathematical commands
- **Graphical User Interface Builder –** will create a graphical design rather than working with source code
- **Translator –** which compiles or interprets the code

Exam Tip

- You could be asked to **Identify** or **Describe** different features within an Integrated Development Environment (IDE) this number could range from 2–5