Learning Aims

Functions of an operating system

- Understand the function and purpose of an operating system
- Describe memory management (paging, segmentation and virtual memory)
- Describe the role of interrupts
- Describe the role of an Interrupt Service Routine (ISR) within the fetch-decode-execute cycle
- Describe the need for processor scheduling algorithms
- Describe scheduling algorithms: round robin, first come first served, multi-level feedback queues, shortest job first and shortest remaining time

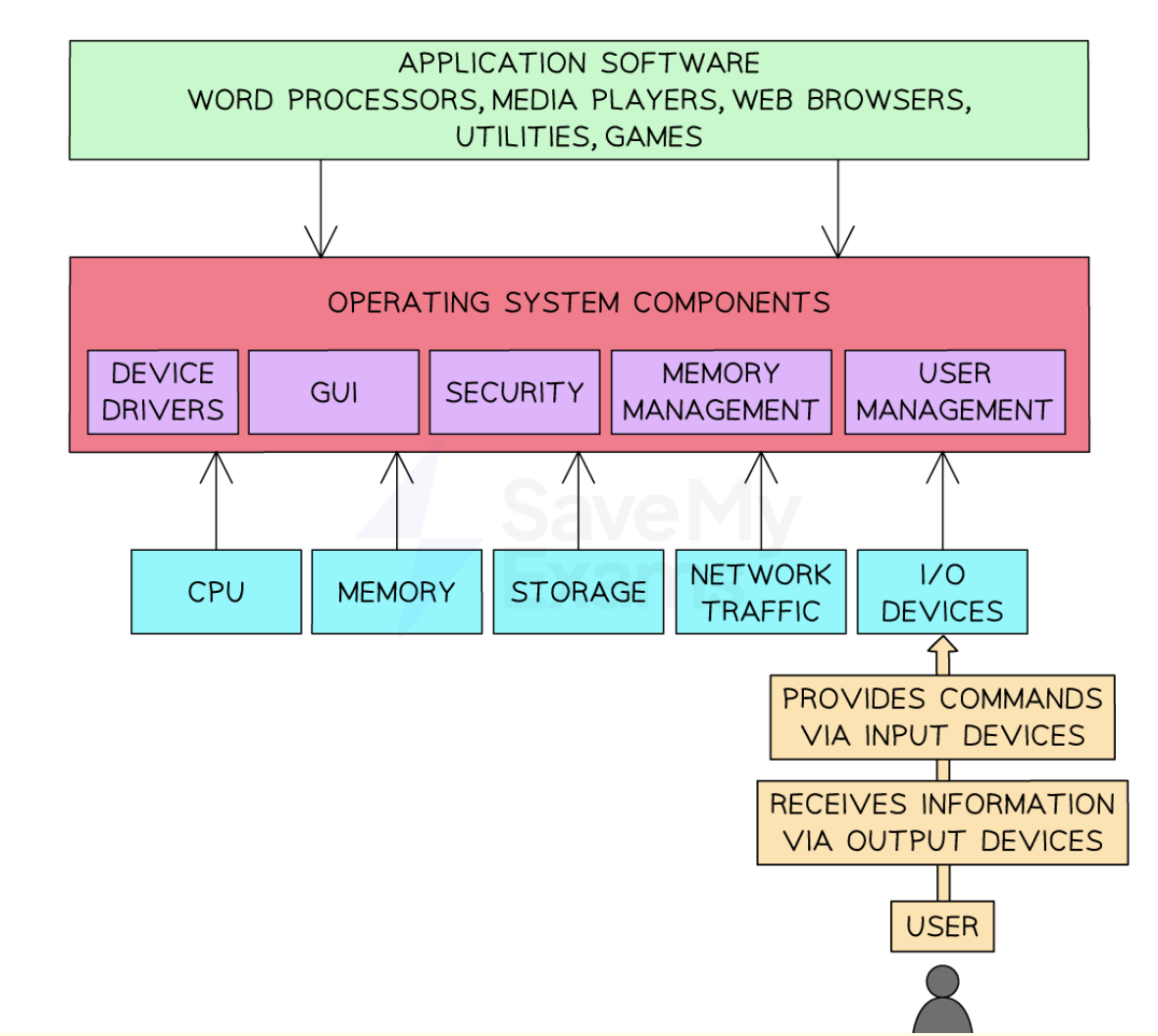
Operating system

The OS is a set of programs that lies between applications software and the computer hardware and has many different functions, including:

- **resource management** – managing all the computer hardware including the CPU, memory, disk drives, keyboard, monitor, printer and other peripheral devices
- **provision of a user interface** (e.g. Windows) to enable users to perform tasks such as running application software, changing settings on the computer, downloading and installing new software, etc.

What is an operating system?

- An **operating system** is **a program or set of programs** that manages the **operations** of the computer for the user.
- It acts as a bridge between the user and the computer's hardware, since a user cannot communicate with hardware directly.
- The operating system is held in permanent storage, for example on a hard disk.
- A small program called the **loader** is held in ROM.
- When a computer is switched on, the loader in ROM sends instructions to load the operating system by copying it from storage into RAM.



Functions of an operating system

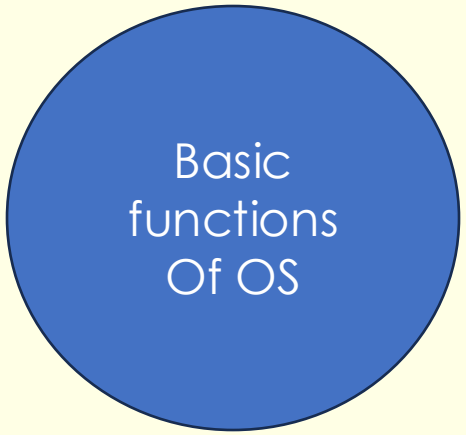
- Resource management
- File management
- Interrupt handling
- Security
- Providing a platform for software to run
- Providing a user interface
- Providing utilities

The basic functions of an operating system include the following:

Resource/memory management
- Moving data between RAM and secondary storage/ virtual memory // paging and/or segmentation
- Allocating/deallocating memory

File management
- Storing files in secondary storage
- Searching for //copying // moving // renaming files/folders

Manage hardware/peripherals
- Tracking all devices connected to the system
- Device drivers



Providing a platform on which to run software
- Allows additional software to be installed on the computer
- To allow the user to complete additional tasks

Security/user management
- Controlling who can access the system //Managing user profiles
- Controlling who can access certain resources on the system // Managing access rights

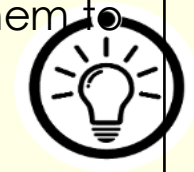
Provide a user interface
- Allowing the user to interact with the software/hardware/computer

Providing utilities
- Used to monitor // manage // maintain the computer
- To manage the security

Resource management	• Operating systems manage the computer's resources, including the CPU, memory, disk drives, and printers
	• They allocate resources to specific tasks and ensure that they are used effectively
	• e.g. when a user opens multiple applications simultaneously, the operating system decides:
	  • **How much memory to allocate to each application**
	  • **When and for how long each gets to use the CPU**
	  • **How to handle data being read from or written to the hard drive.**

File management	• Operating systems handle the storage, retrieval, and manipulation of data files • When working with files, operating systems provide a GUI of the file system that allows a user to decide which directory a file should be saved in and what the file name will be
Interrupt handling	• Interrupt events require the immediate attention of the central processing unit • In order to maintain the smooth running of the system, interrupts need to be handled and processed in a timely manner • E.g. if a user clicks cancel on a file conversion process, a signal is sent from the mouse, interrupts the processor, and the operating system will trigger the cancellation routine
Security	• Operating systems provide various security features such as password-protected system accounts, a **firewall**, virus scanning and **file encryption** • Password-protected system accounts are a very common feature in operating systems • System accounts can also be restricted from performing certain actions, e.g. editing network settings, installing unapproved software, changing the account settings of other users

Providing a platform for software to run	• Operating systems provide a platform on which application software can run, this is mainly by allowing software access to system resources • e.g. if a computer game has intensive graphics and online play, the operating system will grant it access to the GPU and the network card
Providing a user interface	• Operating systems provide interaction in 2 ways: visually through a graphical user interface (GUI) or text-based through a command-line interface (CLI) • Most modern PC operating systems provide both options, and a user will prefer one over the other depending on the task • Mobile operating systems such as Android and iOS provide GUIs that are suitable for interaction through touch • Ubuntu is an OS popular with software engineers because it provides a no-frills GUI and an efficient CLI
Providing utilities	• Utility programs help with system maintenance and security • Some utility programs include: file encryption, file compression, disk defragmentation, system backup, disk cleanup • File encryption allows users to send files over networks securely • File compression reduces the size of a file, which helps send large files over a network • Disk defragmentation physically reorganises files on the hard disk so they can be found and accessed faster • Disk cleanup scans the hard disk for duplicate and corrupt files and deletes them to create more space on the disk • Backup software allows users to restore their system to a point in history



What is an OS?

- An operating system (OS) is a fundamental software that **manages the computer hardware**, **provides common services** for computer programs, and a**cts as an interface** between users and the machine
- There are several different types of operating systems, each with **unique characteristics** and **purposes**
- Understanding these types is essential for both **developing** and **using** computer systems

What are the different types of operating systems?

Type	Description
Distributed Operating Systems	Run on multiple machines appearing as a single unit, used for efficient task distribution and load balancing
Embedded Operating Systems	Designed for specific tasks, is the system running inside a device that is not primarily a computer system, e.g. microwave, dishwasher, washing machine
Multi-tasking Operating Systems	Allows multiple tasks to run concurrently on a single processor, manages system resources and allocates CPU time to different processes
Multi-user Operating Systems	Supports multiple users accessing computer resources concurrently, efficiently manages resource allocation, and provides features for data security and user privacy
Real-Time Operating Systems (RTOS)	Designed for immediate data processing, and can ensure tasks are processed in specific timeframes, highly efficient

Examples

Type	Example
Distributed Operating Systems	Hadoop is an **open-source** OS designed to process big data using multiple **nodes** in a distributed network.
Embedded Operating Systems	**IoT** devices and many household devices contain embedded OS. These devices don't typically run a well-known OS. They run a proprietary OS that has a simple set of functions.
Multi-Tasking Operating Systems	Windows, MacOS, and Linux are multi-tasking OS that can run multiple applications simultaneously.
Multi-User Operating Systems	Windows, MacOS, and Linux are multi-user OSs where multiple users can log in and run independent processes.
Real-Time Operating Systems	Real-time OS are used in industries like aerospace and automotive where low **latency** is critical to safety.

Worked Example

A taxi firm is investigating replacing its drivers with self-driving cars.
Explain why the self-driving system will use a real-time operating system.

[3]

How to answer this question:

- Be able to state the purpose of an operating system and recall the advantages of a real-time operating system
- Link the advantage to the scenario in the question

Example answer that gets full marks:

A self-driving car system must process data in real-time from input sensors such as radar cameras. Data needs to be processed instantly so that the vehicle can operate safely. A vehicle that cannot process data quickly will be an unreliable danger to people. Faster processing of input data will lead to safer operation of the vehicle. A real-time system will replicate hazard perception, navigation, and vehicle control that humans can instinctively do.

Acceptable answers you could have given instead:

- An operating system needs to process tasks quickly. A real-time operating system needs to process them instantly. A real-time operating system should be used in a self-driving car so that it can react to hazards quickly.

Worked Example

A company releases an in-home virtual assistant called 'Bertie Butler'. When placed in a room, the device listens out for the phrase "Hey Bertie". When someone says that phrase, it listens to the following question and tries to give a relevant answer.

The Bertie Butler device runs off an embedded operating system.

Define the term 'embedded operating system'.

[2]

How to answer this question:

- Recall the features of an embedded system
- Use the scenario to include examples in your answer

Example answer that gets full marks:

Embedded systems are those running inside everyday items that are not primarily computers. Embedded systems usually have a simple set of features that they perform efficiently. Some other embedded systems, such as microwaves, have much simpler features and functions.

Acceptable answers you could have given instead:

- An embedded system runs in most household devices that require some processing e.g. microwaves, dishwashers, electric toothbrushes

What is memory management?

- Memory management is a fundamental role of the operating system, dealing with the **allocation** and **deallocation** of the computer's primary memory
- When a user opens an application, its **data is loaded from storage** into active memory so that it can run **smoothly**
- When a user opens a file from the file system, e.g. word document, the CPU loads this **file data,** as well as **application data,** into the primary memory
- Primary memory is a **limited resource** in the system, so it needs **careful management**
- **Benefits** of memory management are:
    - Efficient allocation of memory **enables multitasking**, allowing multiple programs to run at once
    - Memory management **maintains security**, it does not let programs access memory reserved for other programs
- Memory management is made **more efficient** through 3 techniques:
    - **Paging**
    - **Segmentation**
    - **Virtual Memory**

What are interrupts?

- An interrupt is a signal to the processor that **stops its current task** and performs a different task temporarily
- Interrupts can be hardware events or time-sensitive tasks
- When an interrupt occurs, the processor **suspends the current program** execution and transfers control to an **interrupt service routine**

Purpose and role of interrupts

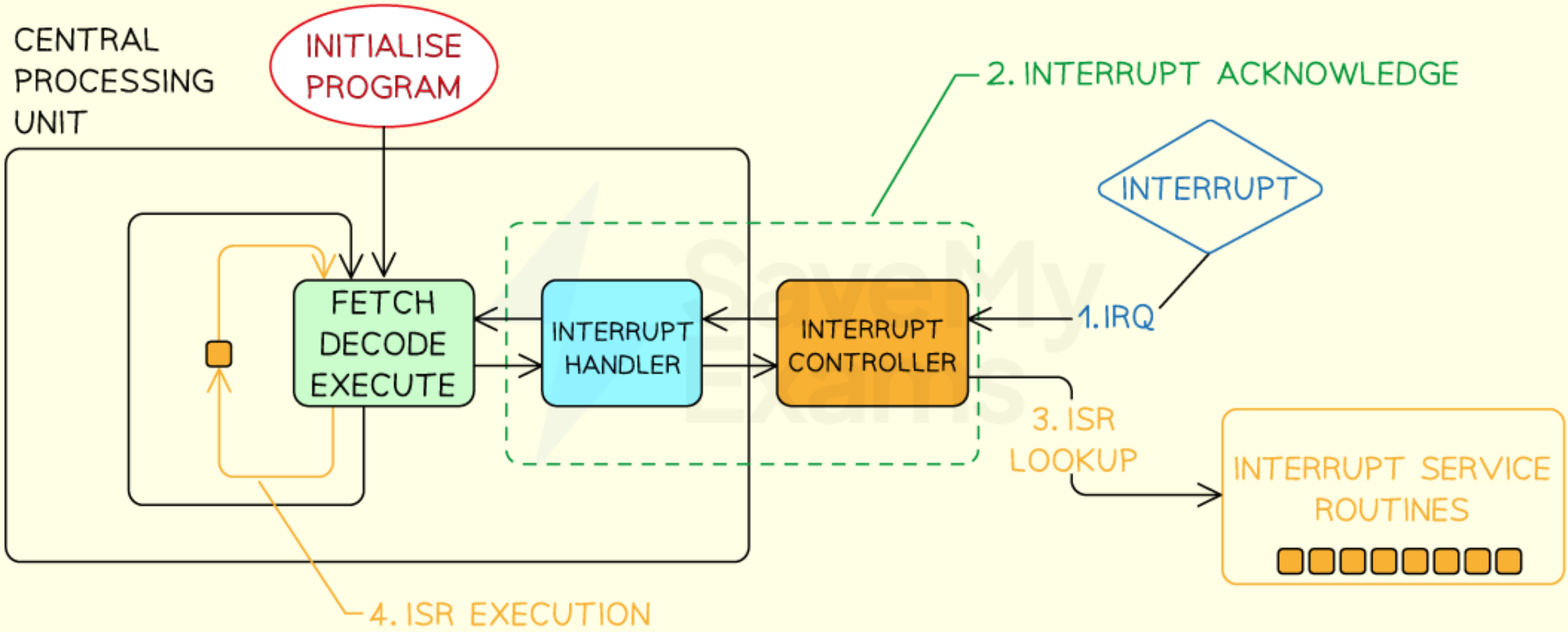
- **Real-time Event Handling**: hardware errors and signals from input devices e.g. hard disk failure
- **Device Communication**: alerts from external devices e.g. printer jams and network errors
- **Multitasking**: suspending processing in one application so that the user can switch to another

Types of interrupts

Type	Definition	Example
Hardware Interrupts	Generated by external devices	Keyboard input, mouse movements, disk I/O requests
Software Interrupts	Triggered by software or the operating system	Application requests to open a file, **division by zero** errors
Trap Interrupts	Intentionally triggered by a program	Software debugging, handling unexpected error cases

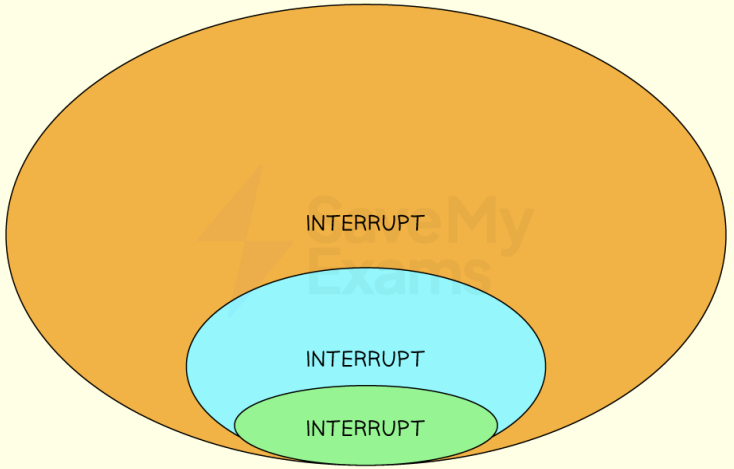
The interrupt process

1. **Interrupt Request (IRQ)**
    1. An external device or software generates an interrupt, signalling the processor to stop its current task
    2. The interrupt controller passes this to the interrupt handler for assessment
2. **Interrupt Acknowledge**
    1. The interrupt handler decides if the interrupt needs to be dealt with now or later
    2. If yes, the **current contents of the processor registers are saved** in memory
3. **Interrupt Service Routine (ISR) Lookup**
    1. The processor fetches the corresponding Interrupt Service Routine (ISR) associated with the interrupt type
4. **ISR Execution**
    1. The processor transfers control to the ISR and executes the routine to handle the specific interrupt
5. **Interrupt Exit**
    1. After the ISR completes, **the processor restores the content of the registers** from step 2
    2. The fetch-decode-execute cycle is resumed



What is an ISR?

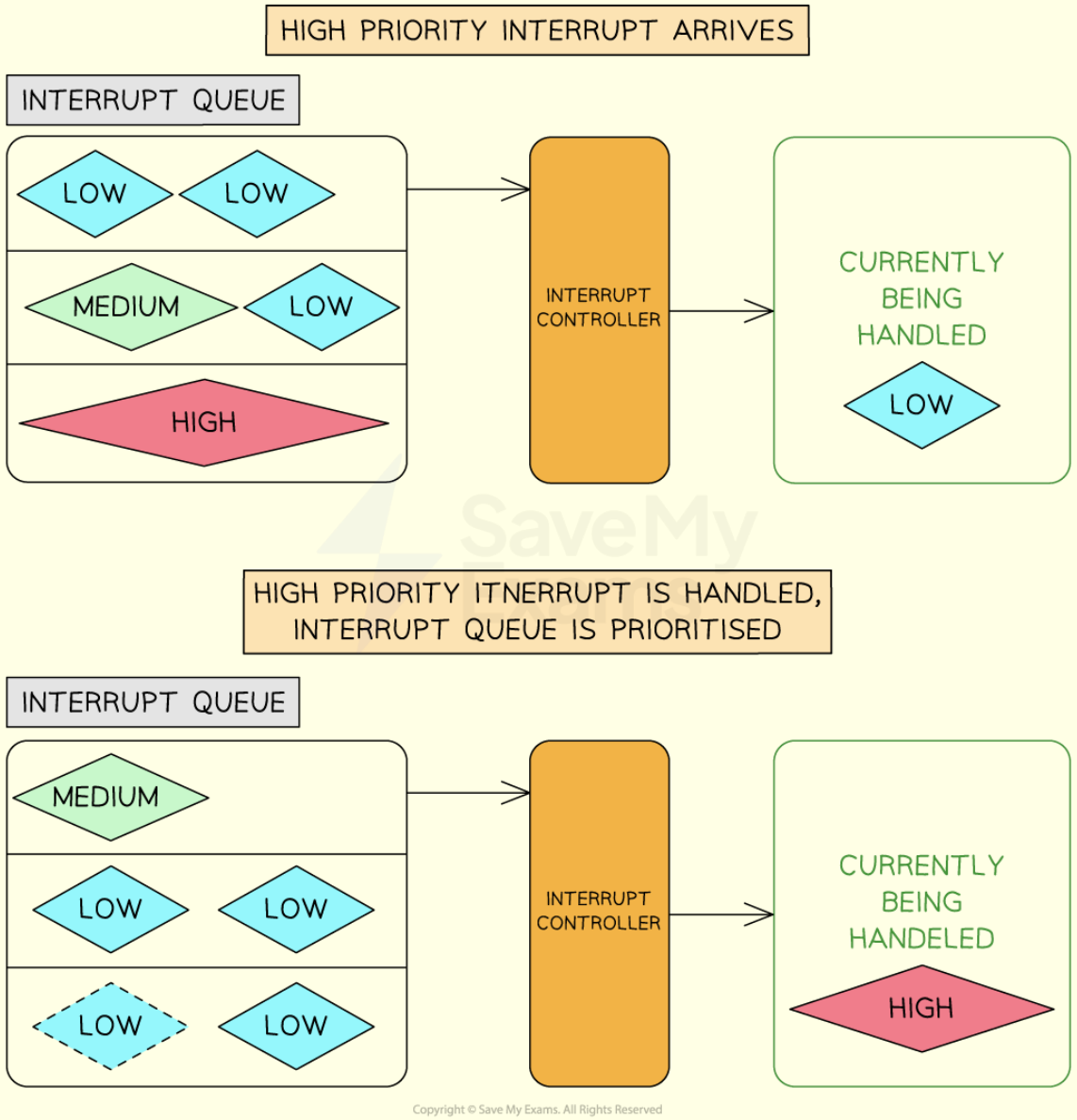
- An ISR is a **special function** that handles a particular interrupt type
- Each type of interrupt has a corresponding routine, e.g. printer jam, hard disk failure, file download error, network connection error all have routines to be followed when they happen
- ISRs should be **concise**, **efficient**, and carefully designed to minimise the time taken to execute, as they often need to handle time-sensitive events



Interrupt priority and nesting

- Interrupt prioritisation means the processor can acknowledge and **switch to resolving a higher-priority interrupt**
- Prioritising interrupts is vital because many things can go wrong at the same time
- **Lower-priority ISRs may be temporarily suspended** until the higher-priority ISR completes the execution
- Nesting of interrupts refers to the ability of the processor to handle interrupts within interrupts
- Proper management of nested interrupts avoids potential conflicts and ensures system stability

Interrupt priority handling



Interrupt sequence

1. Interrupt checked for at start/end of each fetch-execute cycle
2. If the interrupt is of a lower/equal priority to the current process then the current process continues
3. (If interrupt raised) contents of **registers** copied to stack
4. Flags are set to determine if interrupts are enabled / disabled
5. Program counter changed to point to Interrupt Service Routine (ISR) // ISR runs
6. After interrupt complete, previous **register** values restored back from stack
7. Flag is reset
8. If higher priority interrupt received during servicing of interrupt…
9. …this is added to stack and new interrupt dealt with

Scheduling

What is scheduling?

- Deciding **which tasks** to process, for **how long**, and in **what order** is achieved through scheduling algorithms
- A CPU is responsible for processing tasks as fast as possible
- Different algorithms are used to prioritise and process tasks that need CPU time
- The algorithms have different uses, benefits and drawbacks.

Scheduling categories

- **Pre-emptive**: allocates the CPU for **time-limited slots**
- **Non-pre-emptive**: allocates the CPU to tasks for **unlimited time slots**

Pre-emptive Scheduling

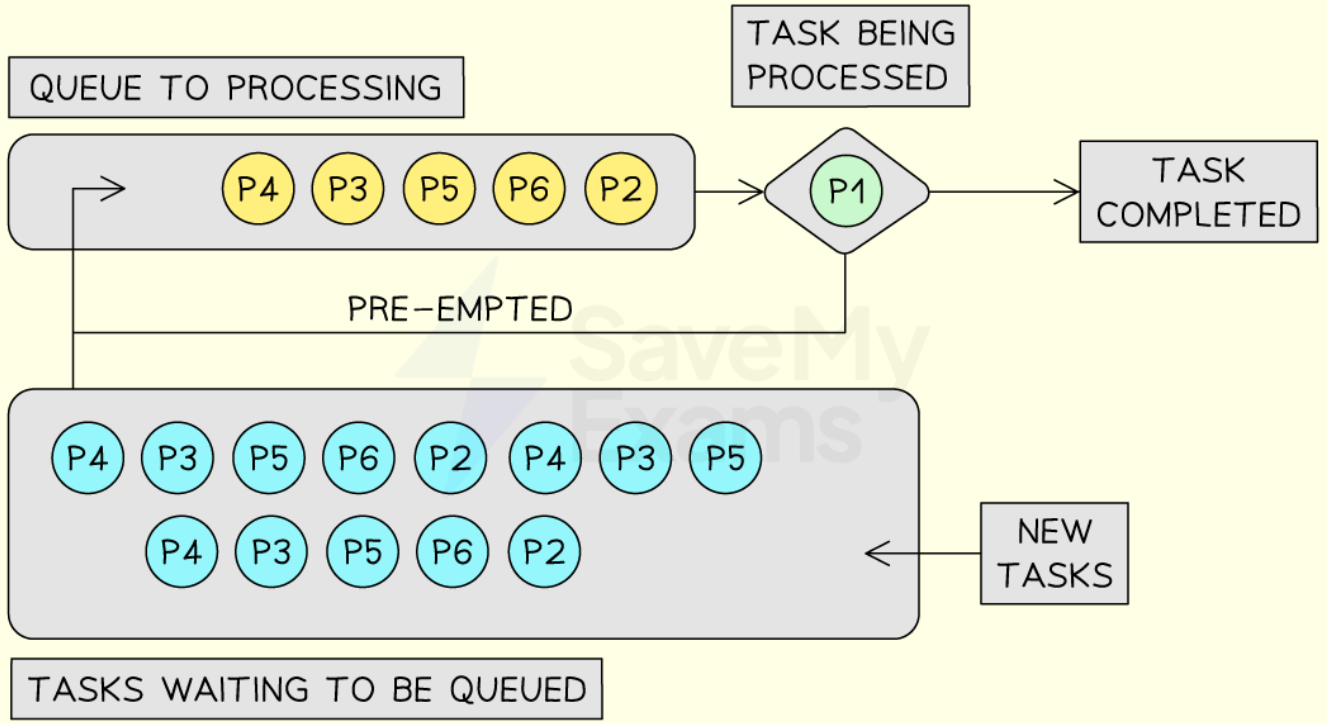
- Allocates the CPU for a specific **time quantum** to a process
- Allows interruption of processes currently being handled
- It can result in low-priority processes being neglected if high-priority processes arrive frequently
- Example algorithms include **Round Robin** and **Shortest Remaining Time First**

Non-pre-emptive scheduling

- Once the CPU is allocated to a process, the process holds it until it completes its **burst time** or switches to a 'waiting' state
- A process cannot be interrupted unless it completes or its burst time is reached
- If a process with a long burst time is running, shorter processes will be neglected
- Example algorithms include **First Come First Serve** and **Shortest Job First**

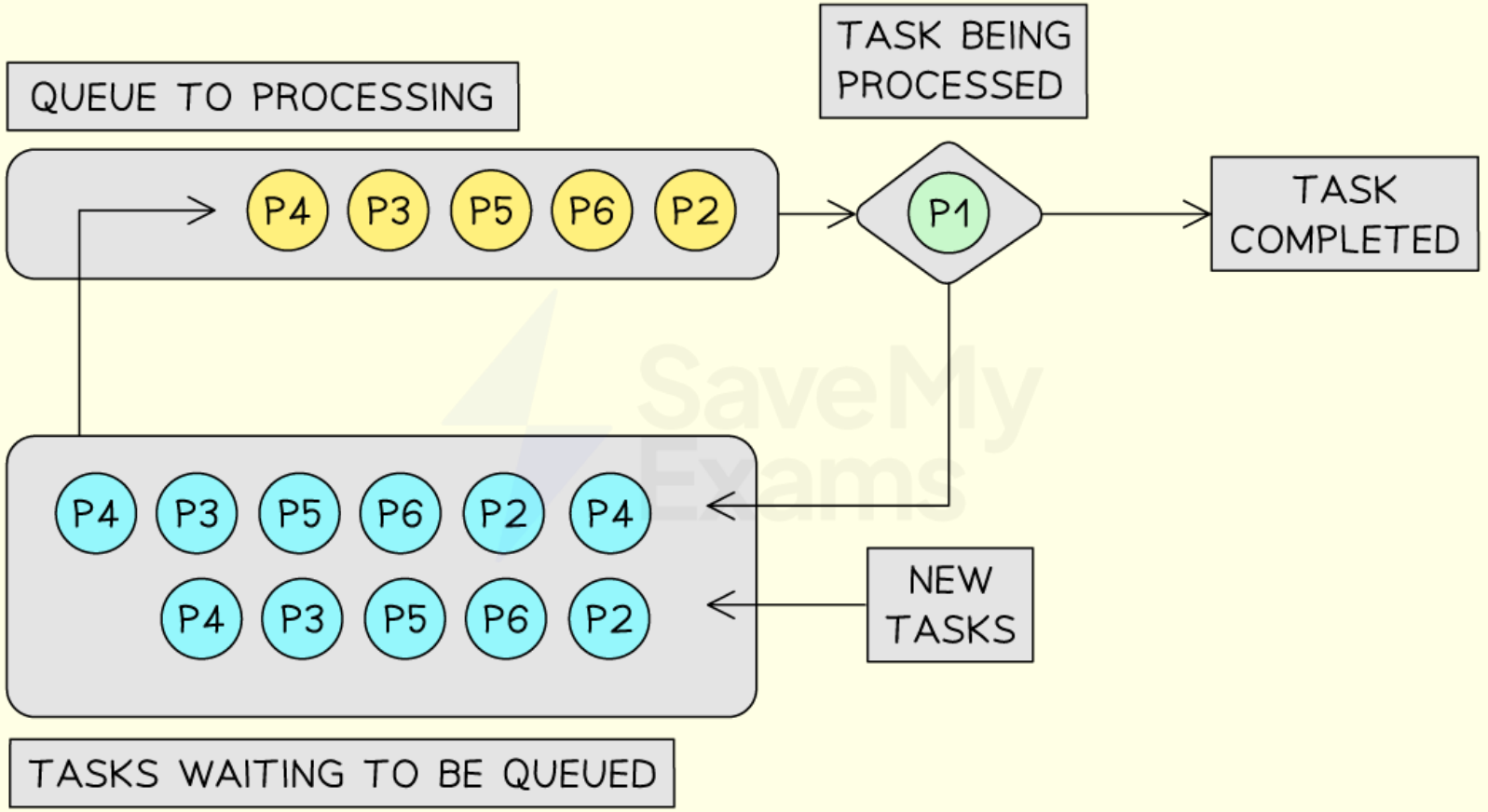
Round robin (RR)

- RR is a pre-emptive scheduling algorithm
- Equally distributing processor time amongst all processes
- Each process is given a **time quantum** to execute
- Processes that are ready to be worked on get queued
- If a process hasn't been completed by the end of its time quantum, it will be moved to the back of the queue



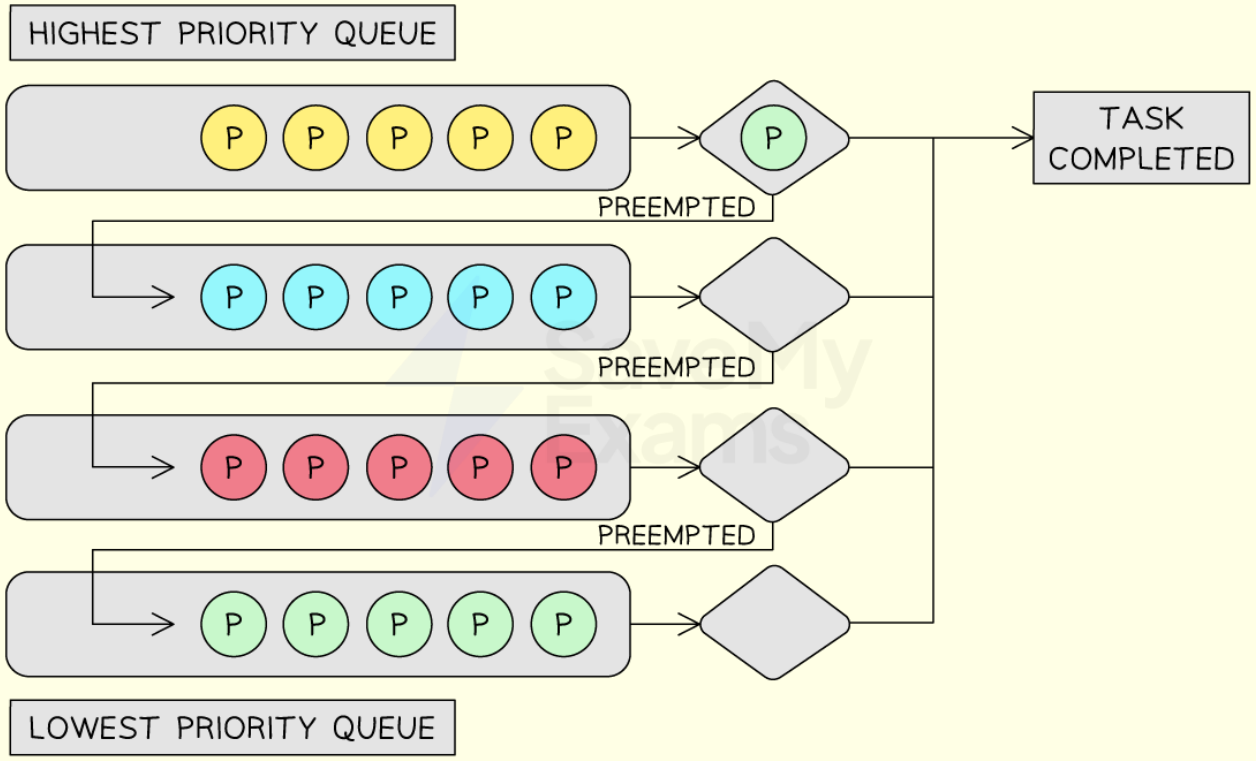
First-Come-First-Served (FCFS)

- FCFS is non-preemptive, prioritising processes that arrive at the queue first
- The process currently being worked on will block all other processes until it is complete
- All new tasks join the back of the queue



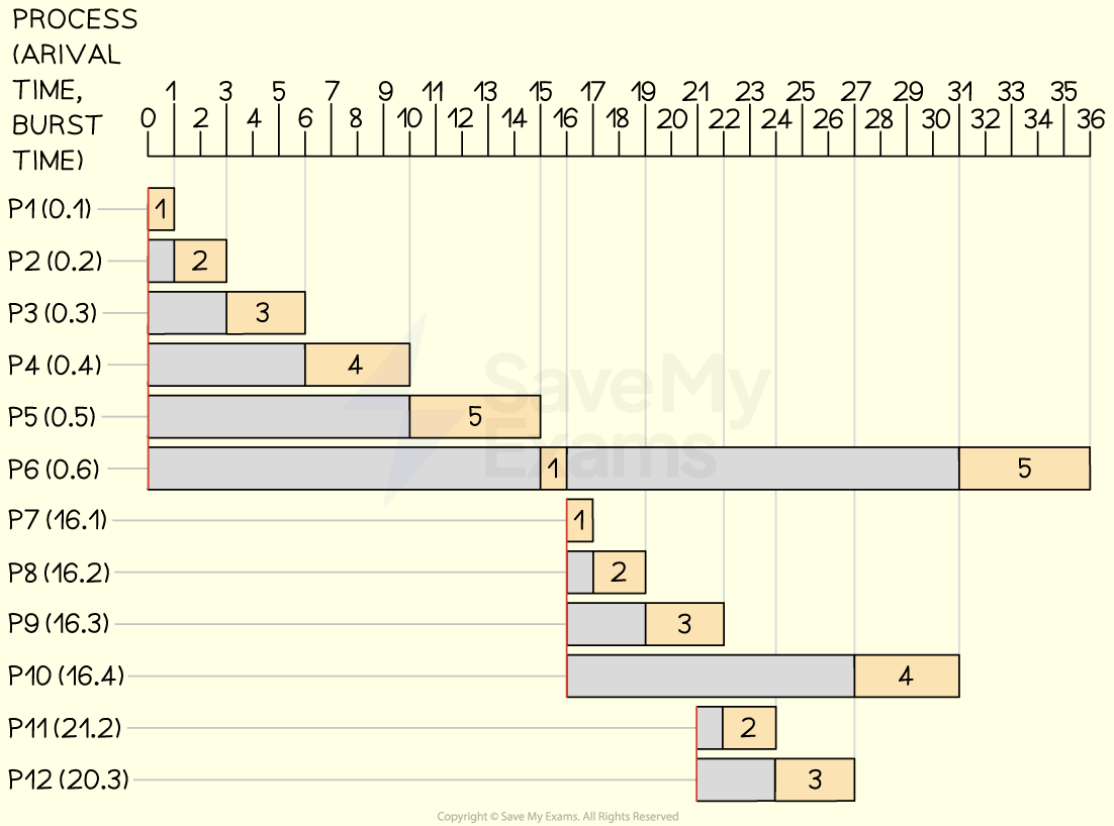
Multi-Level Feedback Queue (MLFQ)

- MLFQ is a pre-emptive priority algorithm where shorter and more critical tasks are processed first
- Multiple queues are used so that tasks of equal size are grouped together
- All processes will join the highest priority queue but will trickle down to lower priority queues if they exceed the time quantum



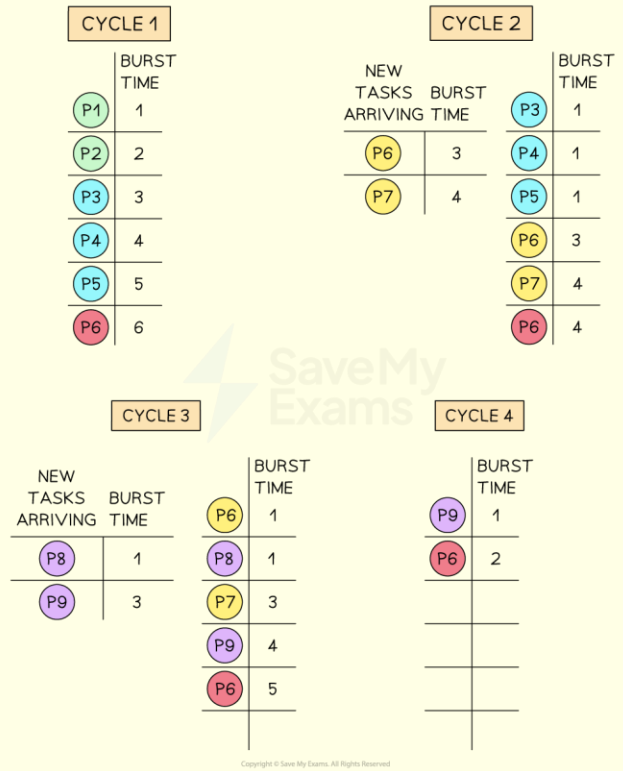
Shortest job first (SJF)

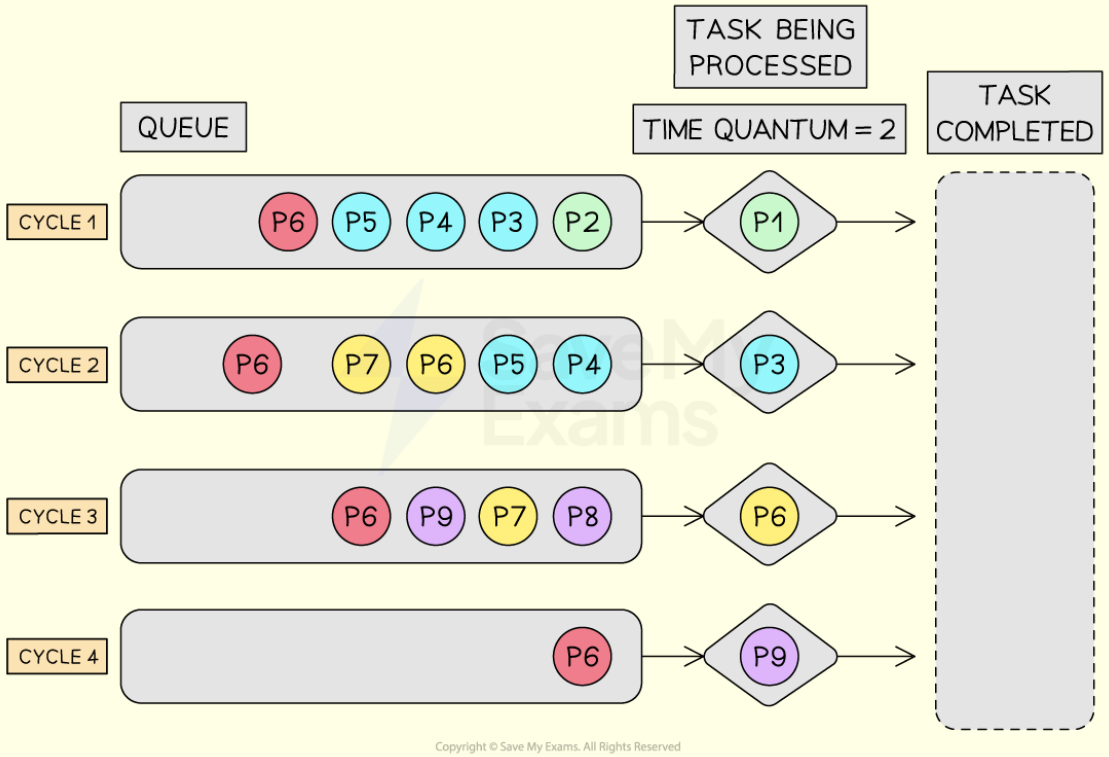
- SJF is non-preemptive, where all processes are continuously sorted by burst time from shortest to longest
- When new processes arrive on the queue, they are prioritised based on their burst time in the next cycle
- Shorter jobs are placed at the front of the priority queue
- Longer jobs have lower priority, so they are placed at the back



Shortest remaining time first (SRTF)

- SRTF is a pre-emptive version of SJF, where processes with the shortest remaining time are higher priority
- Time quantum is set, and if a task doesn't complete in time, it will be re-queued for further processing
- Before the next cycle starts, all processes are inspected and ordered by the shortest remaining time to complete





Comparison and summary of scheduling algorithms

Algorithm	Benefits	Drawbacks
Round Robin	All processes get a fair share of the CPU Good for time-sharing systems Predictable, as every process gets equal time	Choosing the right time quantum can be difficult This can lead to a high turnaround time and waiting time for long processes
First Come, First Served	Simple and easy to understand Fair in the sense that processes are served in the order they arrive	This can lead to poor performance if a long process arrives before shorter processes High-priority tasks wait for their turn in the queue
Multi-Level Feedback Queues	Smaller tasks are prioritised Creates a prioritisation system where similar-sized tasks are queued together	More complex than other algorithms Setting the correct parameters (e.g., number of queues, ageing rules) can be complex
Shortest Job First	Minimises waiting time Efficient and fast for short processes	Requires knowing the burst time of processes in advance Long processes can starve if short processes keep arriving
Shortest Time Remaining	Ideal for jobs that have shorter burst times It is preemptive, so it can be aligned with CPU for best performance (time quantum)	Like SJF, it requires knowing the burst time of processes in advance High context switching overhead due to preemption

- The suitability of a scheduling algorithm largely depends on the specific scenario and the system requirements
- A drawback in one scenario may not be a drawback in another

Worked Example

A company makes anti-virus software. When running anti-virus software, an operating system uses a scheduling algorithm to allocate CPU time to the anti-virus software.

Explain why a First Come First Served scheduling algorithm would not be suitable in this situation.

[2]

How to answer this question:

- Think of the conditions that anti-virus software runs optimally
- Recall the way the FCFS algorithm works and its benefits and drawbacks
- Link how the optimal running of anti-virus is incompatible with FCFS scheduling

Example answer that gets full marks:

Anti-virus software is high-priority because it scans the operating system constantly, looking for threats. When a threat is detected, anti-virus will quarantine or eliminate them. To work effectively, anti-virus software needs high-priority access to CPU time.

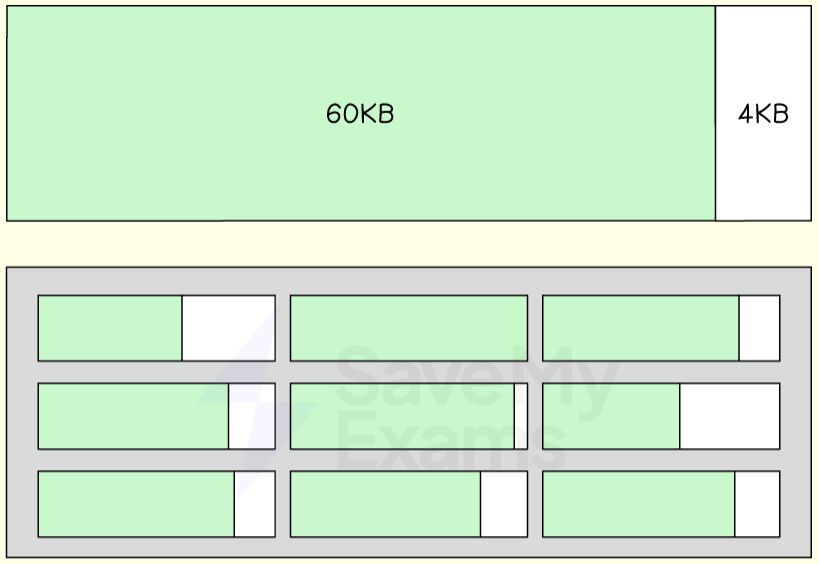
Using FCFS could delay these critical tasks if many other processes are in the queue ahead of the anti-virus software. Other less crucial tasks could get CPU time before the anti-virus process, leading to potential security risks.

Acceptable answers you could have given instead:

- The FCFS algorithm is unsuitable because essential antivirus processing would be placed at the back of the queue and wait for its turn. Lower-priority tasks would use valuable CPU time, meaning the system could be at risk.

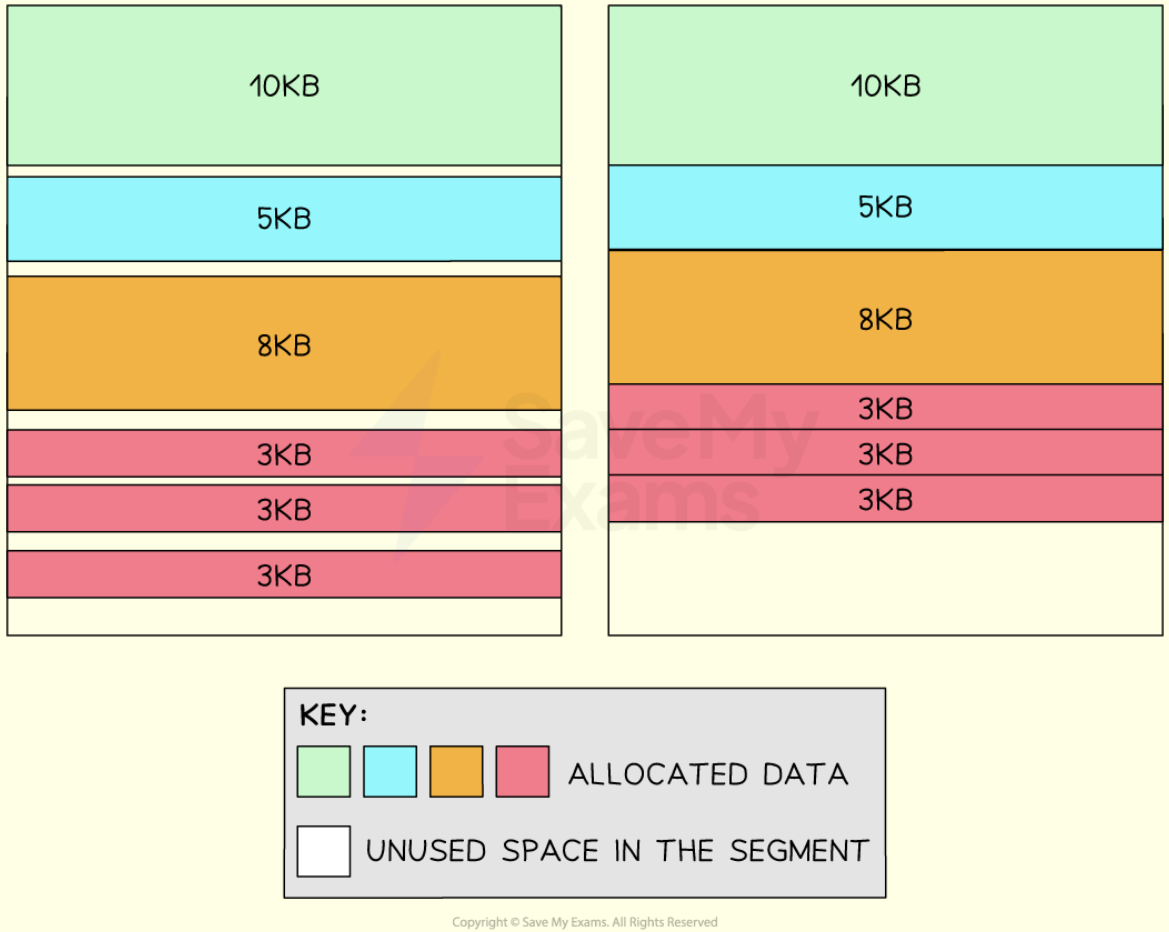
What is paging?

- Paging is a method of **chunking** the primary memory **into equal-sized blocks**
- Data stored in memory will lead to the smooth running of applications
- When an application is launched, data will be moved from the hard disk into Pages for faster access
- As users move between applications, memory is dynamically allocated
- Pages will be taken away from applications not in active use and granted to applications that are in active use
- Paging can lead to **internal fragmentation**
- If a 200KB file is divided into four 64KB Pages, the last Page would have 56KB of unused space
    - **First 64KB** → Full page (64KB used, 0KB unused)
    - **Second 64KB** → Full page (64KB used, 0KB unused)
    - **Third 64KB** → Full page (64KB used, 0KB unused)
    - **Fourth 64KB** → Partial page (**8KB** used, **56KB unused)**
- Unused space in a Page is wasteful because other unrelated data cannot be stored in this Page
- Over time, more pockets of wasted space will exist across the memory; this process is called **internal fragmentation**
- The image below shows a single 64KB Page with 4KB of unoccupied space
- The box below this shows many Pages, each with varying sizes of internal fragments



What is segmentation?

- Segmentation is a method of **chunking** memory into **blocks that correspond to different types of data** needed by an application
- A video editing application may have a Segment for video data, audio data and special effects
- Segments are **not all the same size**; they are sized depending on their allocated data
- Segmentation is space-efficient due to only allocating space depending on the amount an application needs
- Segmentation can lead to **external fragmentation**
- As Segments fill up the memory, physical gaps reduce the maximum size of new Segments that can be allocated
- Below (left) shows different application data assigned to a Segment
- The arrangement of data in the segment **becomes more fragmented over time** because as blocks are taken away it's not possible to guarantee a new block will occupy the same amount of space
- Below (right) shows a defragmented version of the Segment to highlight the total unused space



What is virtual memory?

- If a computer is **running low** on primary memory, it can make secondary storage act as an 'extension' of the main memory
- The operating system can **offload data** from the primary memory into virtual memory
- Virtual memory creates **an illusion of a larger memory** and enables applications to continue to multitask
- However, accessing data in virtual memory is **considerably slower** compared to RAM
- **Solid-state drives** are faster than traditional hard-disk drives, but **neither are as fast** as RAM
- **Over-reliance** on virtual memory can lead to **performance issues**

Worked Example

Describe how the operating system would use virtual memory to load program C when there's not enough space in physical memory.

[3]

Answer that gets full marks:

The operating system can use virtual memory to act as an extension of the computer's primary memory. This means less-critical data can be offloaded from the primary memory into virtual storage, useful when a higher-priority set of processes require immediate attention. If the OS offloaded data from the RAM into virtual memory, this would free up space for program C to be loaded into RAM.

Acceptable answers you could have given instead:

- Program C can be loaded into the RAM if the operating system moves files and data into virtual memory. Virtual memory acts as an extension of the RAM.

Worked Example

Imogen buys a desktop computer. It comes with an operating system installed.
Describe two ways that an operating system could manage physical memory.

[4]

How to answer this question

- Recall two methods of memory management (Paging, segmentation, virtual memory)
- Explain how each of these works and how they assist in memory management

Example answer that gets full marks:

Imogen's operating system could manage physical memory in two ways. Paging, which is a method of dividing memory into fixed-size chunks known as pages. Application data can be allocated and deallocated to pages making it a flexible system depending on the activity of the user.

The system could also use virtual memory, a method of extending the available physical memory by using a portion of the hard drive. This allows more programs to run simultaneously and enhances the overall system performance.

Acceptable answers you could have given instead:

Imogen's computer uses paging to manage memory. This means it breaks down the memory into fixed-size pieces and swaps them in and out as needed.

- The computer also uses something called virtual memory. It uses part of the hard drive to act like extra memory, so more programs can run at the same time, and everything works faster.

What does BIOS mean?

- BIOS refers to the Basic Input/Output System of a computer
- BIOS is a piece of **firmware** stored on a small memory chip on the motherboard
- On system start, the BIOS is the first software to run
- It performs a **POST (Power-On Self-Test)**, a diagnostic testing sequence that ensures all the hardware components are working properly
- If the BIOS encounters any errors during this test, it will either halt the boot process or issue an error message
- If the POST succeeds, the BIOS will run the **Bootstrap loading sequence**, which is the program responsible for starting the operating system

Worked Example

In the late 1990s, the CIH virus hit the headlines because it could overwrite and destroy the contents of a computer's BIOS.

Describe the effect of a computer having its BIOS overwritten.

[2]

How to answer this question:

- Explain the role of BIOS in a computer system
- Explain the consequence of having a destroyed BIOS

Example answer that gets full marks:

The BIOS is crucial in initialising the hardware components and operating the system. If a virus, such as the CIH virus, overwrites or destroys the contents of a computer's BIOS, it will be unable to perform the Power-On Self-Test (POST) and fail to load the bootstrap program. This would mean the computer will not be able to initialise the operating system.

Acceptable answers you could have given instead:

- If the BIOS has been overwritten, it will not be able to perform the Power-On Self-Test (POST) and, therefore, unable to load the bootstrap program. This would mean the operating system would fail to start.

What is a Device Driver?

- A device driver is a piece of software that enables communication between an operating system and specific hardware devices such as:
  - Printers
  - Graphics cards
  - Network cards
- Device drivers allow the OS to **control and interact** with those devices
- Because many external devices have embedded system software, a driver **bridges the gap** between a major operating system and a tiny hardware OS
- Device drivers make it possible to **perform specific operations** on the hardware e.g. a printer driver enables the OS to send print commands and manage print jobs
- Most hardware manufacturers write their own device driver software, meaning a single operating system may have **several printer drivers** installed

Worked Example

Adding a DVD drive to a computer often requires the installation of a piece of software called a device driver.

State the purpose of a device driver.

[1]

How to answer this question:

- Pick one example of a hardware device and describe what happens between it and the operating system
- Explain that this is made possible through device driver software

Example answer that gets full marks:

The purpose of a device driver is to enable the operating system to interact with and control a hardware device. For example, a printer would have an associated device driver so the OS could send or receive data from a printer system.

Acceptable answers you could have given instead:

- The purpose of a device driver is to enable communication between an operating system and external hardware.

What are virtual machines?

- Virtual machines (VMs) are entire operating systems **running inside another operating system**
- A user running Windows 11 could run a virtual machine of MacOS
- This would allow them to navigate the GUI of MacOS and install software on it
- Running a virtual machine helps access software that is only designed to run on specific operating systems

Benefits and drawbacks

Benefits	**Drawbacks**
• Virtual machines (VMs) are entire operating systems running inside another operating system • Allows running different operating systems (e.g. Linux on Windows) simultaneously on a single machine. • VMs are a way to create isolated test environments, that leave the host operating system unaffected • Multiple VMs can run on one physical machine, maximising hardware usage and reducing the need for multiple physical devices. • deal for running potentially unsafe software or malware for analysis in a controlled environment.	• VMs are slower than physical machines because resources (CPU, RAM, disk) are shared with the host system and other VMs. • Running multiple VMs can consume significant system resources, leading to slower performance or system crashes if not managed properly

Worked Example

Explain why the programmers of anti-virus software may make use of virtual machines when developing the updates.

[3]

How to answer this question:

- Recall two benefits of using virtual machines in developing software
- Link these benefits to some considerations needed when developing anti-virus software

Example answer that gets full marks:

Virtual machines (VMs) are essential in developing anti-virus software updates for several reasons. VMs create an isolated environment that is separate from the host operating system. This enables developers to safely work with virus test code that could risk the integrity of their own operating system. VMs can also be configured to emulate various types of hardware. This allows developers to understand how their anti-virus software will perform on different devices and under different conditions, ensuring a wider range of compatibility.

Acceptable answers you could have given instead:

- Virtual machines create an isolated test environment from the host operating system. This means working with harmful test code carries less risk. Virtual machine management software can monitor the VM through the Hypervisor. This will show how the software affects VM system performance, allowing the programmer to make changes where needed.