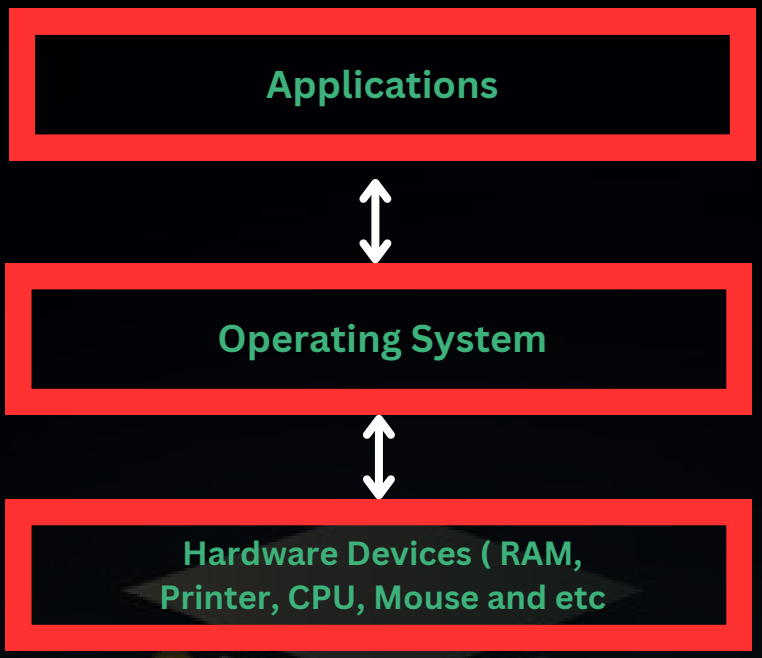
# Purpose Of An Operating System

Papers Dock

COMPUTER SCIENCE 9618 PAPER 3

# Purpose Of An Operating System

**What is an operating system ?**




Provides interface between users and hardware

## Describe the ways in which the user interface hides the complexities of the hardware from the user

- The user interface hides the complexities of the computer hardware and operating system, making it easier for users to interact.
- It provides different access systems to accommodate users with varying needs, ensuring ease of use.
- Complex commands involving memory locations, buses, and hardware operations are avoided, allowing users to perform tasks without technical knowledge.

Example

- Clicking on an icon instead of writing code simplifies interactions.
- Using a graphical user interface (GUI) with icons for navigation makes computing more user-friendly.

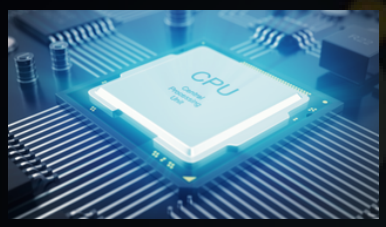
# Resource Management

**Question : What are resources ?**

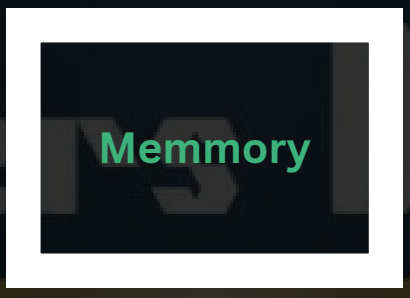
- **C.P.U**
- **Memory**
- **Input / Output devices**

Resource management focuses on utilizing the resources and maximize the use of resources
Deals with input/output operation

# Direct Memory Access









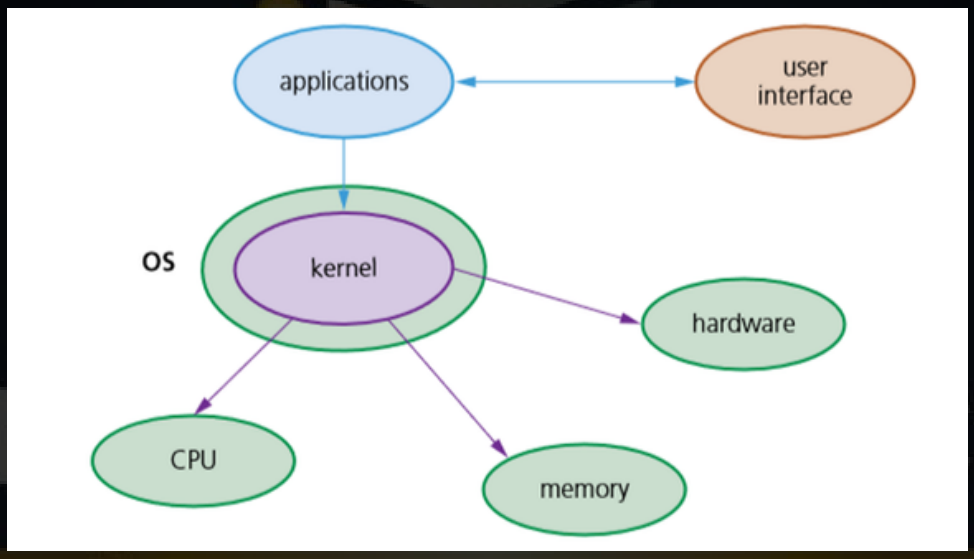

- We use DMA controller to give access to memory directly. It allows the hardware to access the main memory independently of the CPU.
- It frees up the CPU to allow it to carry out the other tasks.

1. **DMA initiates the data transfer**
2. **While CPU carries out other tasks**
3. **Once the data transfer is complete, an interrupt signal is sent to the CPU from the DMA.**

# Kernel

If an application wants to access a hardware component such as flash light so it first goes to kernel and seek permission to use it.

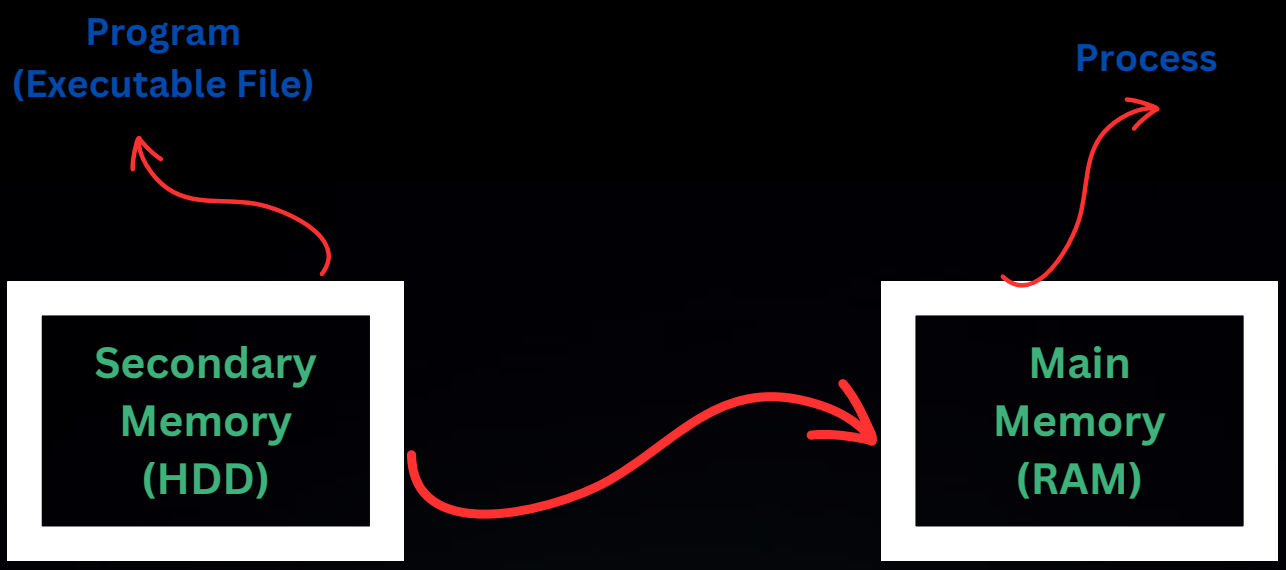
- **Is the part of operating system**
- **Responsible for communicating between hardware, software, and memory.**
- **Responsible for process management, device management, memory management.**




**Question : How the operating system hides the complexities of the hardware from the user?**

- **Operating system provides interface e.g. GUI which helps to use the hardwares**
- **Operating System uses device drivers to synchronize the hardware**

# Process Management




- **Program is the written code**
- **Process is the executing code.**

# Multitasking

Multitasking in an operating system allows a user to perform more than one task at a time.

- **To ensure multitasking operates correctly, scheduling is used to decide which process should be carried out.**
- **Multitasking ensures the best use of computer resources by monitoring each state of process.**
- **It should seem that many processes are executed at the same time.**
- **In fact, Kernel overlaps the execution of each process based on scheduling algorithm.**

- **Preemptive:** Preempt (to take action, to steal). When CPU is allocated to a particular process and if at that time a higher priority process comes, then CPU is allocated to that process.

- **Nonpreemptive:** Does not take any action until the process is terminated.

## Preemptive

- Resources are allocated to a process for a limited time.
- The process can be interrupted while it is running.
- More flexible form of scheduling.

## Non Preemptive

- Once the resources are allocated to a process, the process retains them until it has completed its burst time (amount of time required by a process for executing on CPU).
- The process cannot be interrupted while running (It must finish or switch to waiting state).
- More rigid form of scheduling.

**Question : Explain why an operating system needs to use scheduling algorithms ?**

- To allow multitasking to take place
- To ensure fair usage of processor
- To ensure fair usage of peripherals
- To ensure fair usage of memory
- To ensure higher priority tasks are executed sooner
- To ensure all processes have the opportunity to finish
- To minimize the amount of time users must wait for their results
- To keep CPU busy at all times
- To service the largest possible number of jobs in a given amount of time.

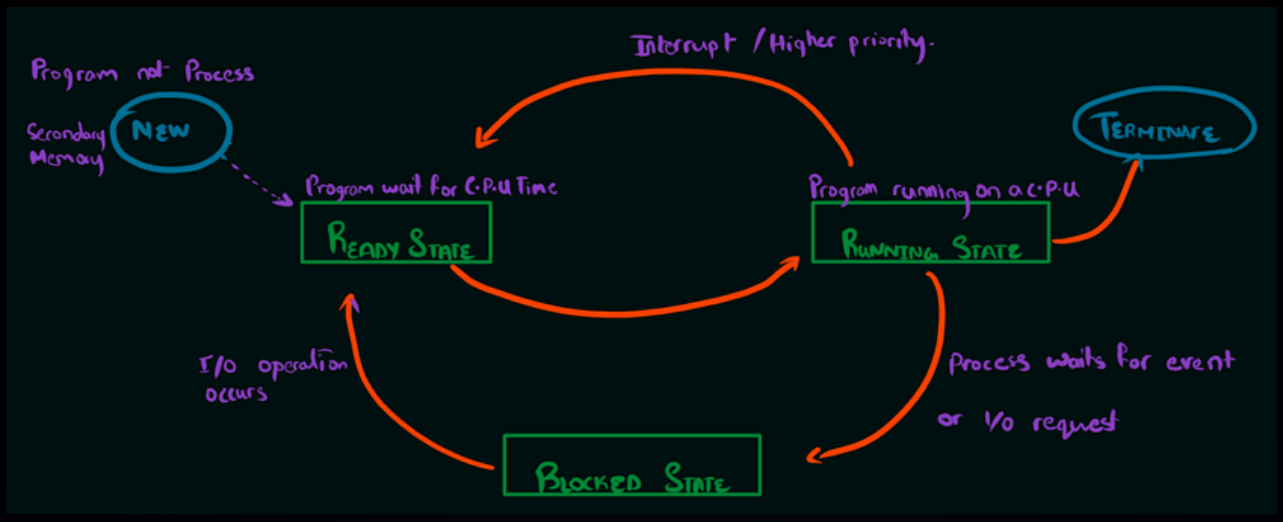
# Process State

Running Ready Blocked

**Case 1 :** No Interrupts / I/O Requests


**Case 2 :** I/O Requests

## Case 3 : Preemptive




### State 1 : Ready

**Description:**

- The process is not being executed
- The process is in the queue
- Waiting for the processor's attention / time slice.

### State 2 : Running

**Description:**

- The process is being executed
- The process is currently using its allocated processor time / time slice.

### State 3 : Blocked

**Description:**

- The process is waiting for an event
- so it cannot be executed at the moment e.g.input/output.

# Conditions For Transition Between State

## READY → RUNNING

- Processor is available, current process no longer running.
- Process was at the head of the ready queue // process has the highest priority.
- OS allocates processor to process so that process can execute.

## RUNNING → READY

- When process is executing, it is allocated a time slice.
- When time slice is completed, interrupt occurs, and process can no longer use processor even though it is capable of further processing.



## RUNNING → BLOCKED

- Process is executing (running state), and when it needs to perform I/O operation, it is placed in blocked state until I/O operation is completed.

**Question : Explain why a process cannot be moved from the blocked state to the running state?**

- When I/O operation is completed for a process in the blocked state,
- The process is transferred to the ready state.
- OS decides which process to allocate to the processor.

**Question : Explain why a process cannot move directly from the ready state to the blocked state?**

- To be in the blocked state, the process must initiate some I/O operation.
- To initiate the operation, the process must be executing.
- If the process is in the ready state, it cannot be executing.

## Scheduler

- High Level Scheduler: Decides which processes are to be loaded from backing store into ready queue.
- Low Level Scheduler: Decides which of the processes in ready state should get use of processor. Or which process is put into running queue based on position or priority.

# Scheduling Routine Algorithms

- First come first served scheduling
- Shortest job first scheduling
- Shortest remaining time first scheduling
- Round Robin

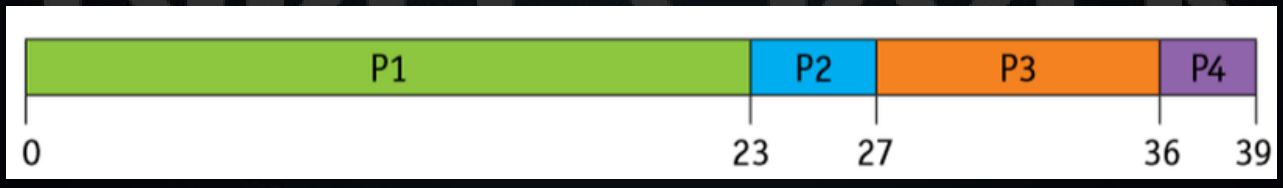
# First Come First Served Scheduling

- Non-preemptive
- Based on arrival time
- Uses first-in first-out principle (FIFO)

| Process | Arrive Sequence | Burst Time |
|---|---|---|
| P1 | 1 | 23 ms |
| P2 | 2 | 4 ms |
| P3 | 3 | 9 ms |
| P4 | 4 | 3 ms |



So the Queue will be



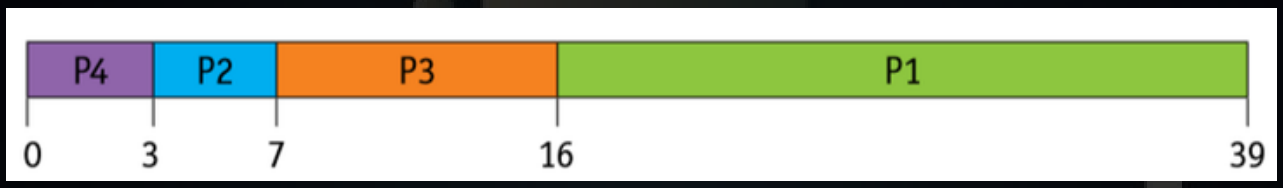
# Shortest Job First Scheduling

- Non-preemptive
- Based on arrival time
- Uses first-in first-out principle (FIFO)

| Process | Arrive Sequence | Burst Time |
|---|---|---|
| P1 | 1 | 23 ms |
| P2 | 2 | 4 ms |
| P3 | 3 | 9 ms |
| P4 | 4 | 3 ms |

with SJB, the process requiring the least CPU time is executed first

So the Queue will be




# Shortest Remaining Time First

- Preemptive
- The processes are placed in ready queue as they arrive
- But when a process with the shortest burst time arrives
- the existing process is removed
- The shorter process is then executed first

| Process | Burst Time (ms) | Arrival Time of Process (ms) |
|---|---|---|
| P1 | 23 | 0 |
| P2 | 4 | 1 |
| P3 | 9 | 2 |
| P4 | 3 | 3 |

# Round Robin

- Preemptive
- A fixed time slice is given to each process; this is known as time quantum.
- The running queue is worked out by giving each process its time slice in the correct order. If a process completes before the end of its time slice, the next process is brought into the ready queue for its time slice.

| Process | Burst Time (ms) | Arrival Time of Process (ms) |
|---|---|---|
| P1 | 23 | 0 |
| P2 | 4 | 1 |
| P3 | 9 | 2 |
| P4 | 3 | 3 |

**Time Quantum = 5ms**

**Ready**

**Running**

**Question : Explain the need for scheduling in process management ?**

- Process scheduling allows more than one task to be executed at the same time which enables multi-tasking
- To allow high-priority jobs to be completed first.
- To keep the CPU busy all the time.
- To ensure that all processes execute efficiently.
- To have reduced wait times for all processes and to ensure all processes have fair access to the CPU

# Description And Benefits And Drawbacks

## First Come First Served (FCFS) Scheduling

**Description:**

- FCFS scheduling processes requests in the order they arrive. Each process is queued as it is received and executed one by one, without preemption.
- It follows a non-preemptive approach, meaning once a process starts execution, it runs until completion.

**Benefits:**

- Simple and easy to implement as it does not require complex scheduling logic.
- Prevents starvation, ensuring every process will eventually get a chance to run.
- Low overhead as there is no need for frequent context switching.

**Drawbacks:**

- Poor response time for long processes, as shorter jobs must wait for long ones to finish.
- Can lead to convoy effect, where short processes are stuck waiting behind long processes.
- Not suitable for time-sharing systems, as it does not allow preemption.

## Shortest Job First (SJF) Scheduling

**Description:**

- SJF scheduling selects the process with the shortest CPU burst time and executes it first.
- It is non-preemptive, meaning once a process starts, it runs to completion before the next shortest process is selected.

**Benefits:**

- Increases throughput, as shorter processes finish quickly, allowing more processes to be executed in less time.
- Minimizes average waiting time, as shorter processes do not have to wait for long-running processes.

**Drawbacks:**

- Can cause starvation, as long-running processes may never get CPU time if short jobs keep arriving.
- Difficult to implement, as it requires knowing the exact burst time of processes beforehand.
- Not ideal for dynamic environments, where process execution times may change.

## Shortest Remaining Time First (SRTF) Scheduling

**Description:**

- Preemptive
- The processes are placed in ready queue as they arrive
- But when a process with the shortest burst time arrives
- the existing process is removed
- The shorter process is then executed first

**Benefits:**

- More responsive than SJF, as shorter jobs can interrupt longer ones.
- Minimizes average turnaround time by prioritizing short tasks.
- Efficient CPU utilization, reducing idle time.

**Drawbacks:**

- Can cause high overhead, as frequent context switching occurs when new shorter processes arrive.
- Starvation of long processes, as they may keep getting preempted by shorter jobs.
- Requires accurate estimation of remaining CPU time, which is difficult in real-world scenarios.

## Round Robin (RR) Scheduling

**Description:**

- RR assigns a fixed time slice (quantum) to each process in a cyclic order.
- If a process does not finish within its time slice, it is preempted and moved to the back of the queue.

**Benefits:**

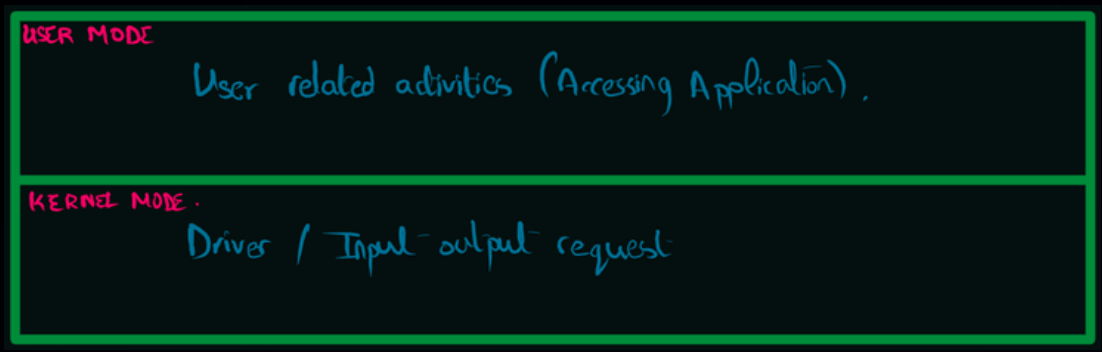
- Ensures fairness, as every process gets CPU time equally.
- Prevents starvation, since all processes receive execution time in each cycle.
- Better responsiveness, making it suitable for time-sharing systems and multi-user environments.

**Drawbacks:**

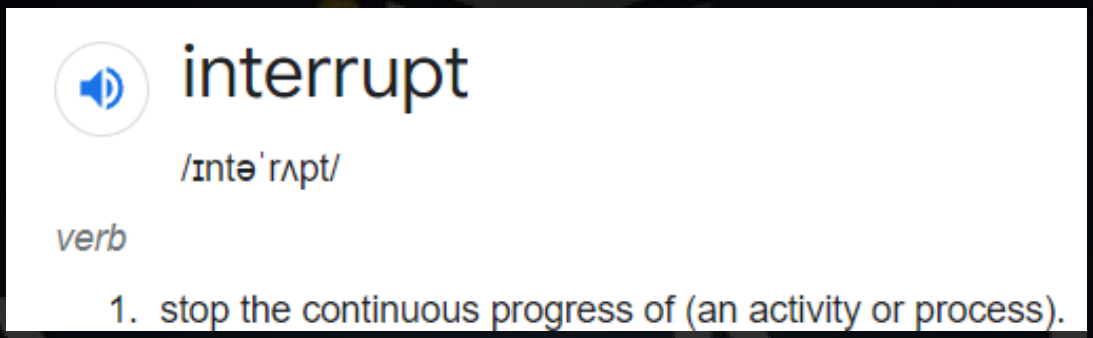
- High context switching overhead, especially if the time slice is too small.
- Inefficient for long processes, as they require multiple cycles to complete.
- Performance depends on time quantum selection – too small leads to excessive switching, too large behaves like FCFS.

# Interrupt Handling

## User Mode And Kernel Mode




**Processor Switches Between User Mode And Kernel Mode**



interrupt

/ɪntəˈrʌpt/

*verb*

1. stop the continuous progress of (an activity or process).

**Interrupt is a kind of signal to OS from the device which is connected to computer. Sometimes interrupts are within the computer.**

The processor will check for interrupt signals and will switch to kernel mode if any of the following type of interrupt signals are sent:

- Device Interrupt (Printer out of paper)
- Exception (Instruction faults such as division by zero)
- Traps / Software Interrupts (Process requesting a resource)

Interrupt Dispatch Table: to determine the current response to interrupt.

Interrupt Priority Level: numbered (0 - 31)

- When an interrupt is received, other interrupts are disabled so that the process that deals with the interrupt cannot itself be interrupted.
- The state of the current task/process is saved on the kernel stack.
- The system now jumps to the interrupt service routine (using the IDT).
- Once completed, the state of the interrupted process is restored using the values stored on the kernel stack, the process then continues.
- After an interrupt has been handled, the interrupt needs to be restored so that any further interrupts can be dealt with.

**Question : How does the kernel of the OS act as an interrupt handler?**

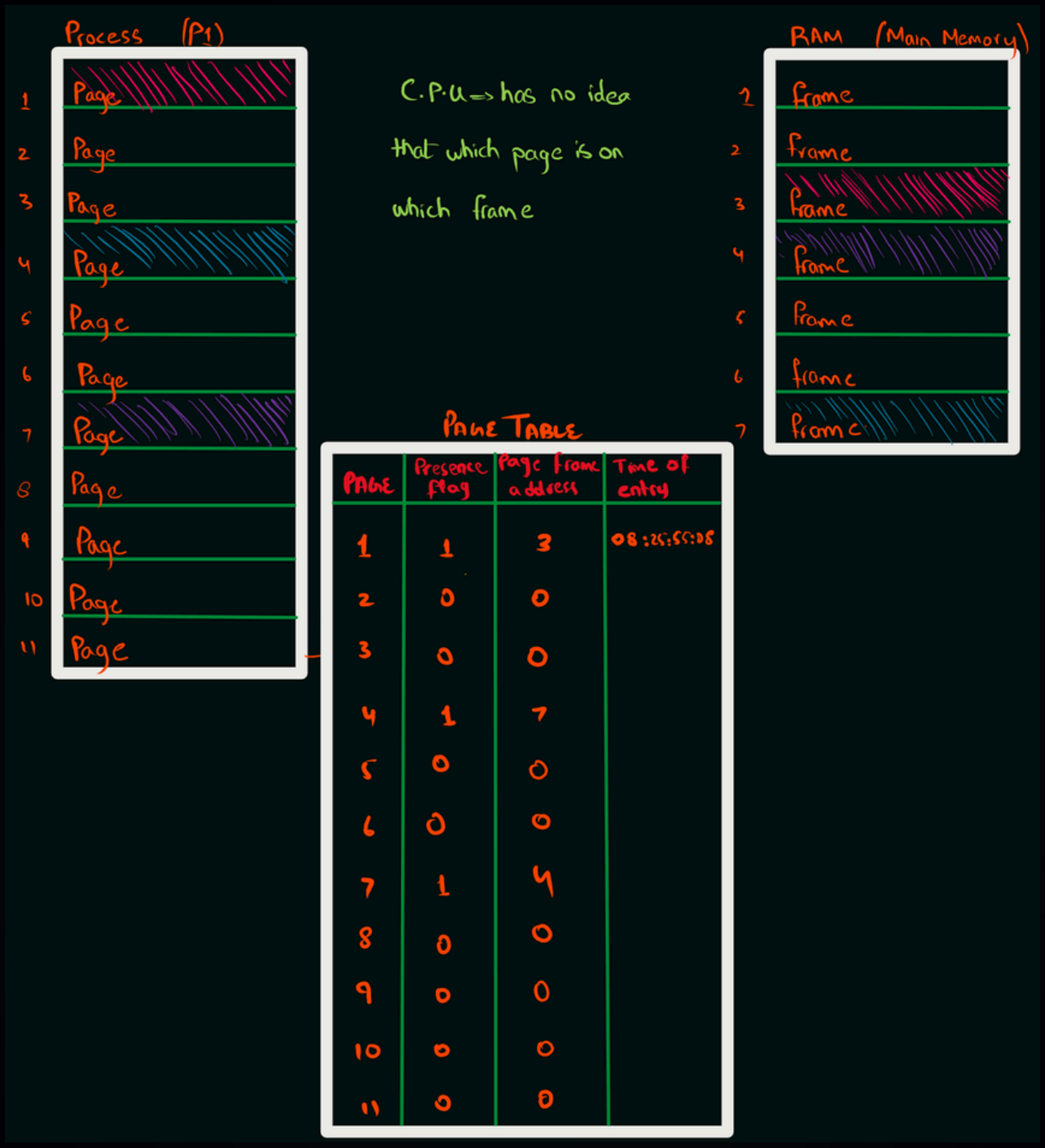
- When an interrupt is received, other interrupts are disabled to ensure that the process handling the interrupt cannot be interrupted itself.
- The state of the current task or process is saved on the kernel stack to preserve its progress.
- The system jumps to the appropriate Interrupt Service Routine (ISR) by looking it up in the Interrupt Descriptor Table (IDT).
- The ISR is executed, handling the specific interrupt event (e.g., hardware or software request).
- Once the interrupt is handled, the state of the previously interrupted process is restored from the kernel stack, allowing it to continue its operation.
- After the interrupt has been fully handled, interrupts are re-enabled so the system can respond to future interrupts.

**Question : How is interrupt handling used to manage low-level scheduling?**

- The system uses regular timer interrupts to manage how long each process can run, which helps with time-sharing in multitasking systems.
- When a timer interrupt occurs, the kernel checks if a higher-priority task is waiting to run and can preempt the current task.
- If needed, the kernel saves the current process's state and switches to a new process (context switch), enabling efficient process scheduling.
- Interrupts allow the system to quickly respond to real-time events (like network requests or hardware I/O), which helps manage process priorities and scheduling in a dynamic environment.

# Memory Management

## Concept Of Paging




**Rest of pages which are not present in main memory they are stored in Hard drive.**

# Page Replacement

Page replacement occurs when a requested page is not in memory (flag = 0). When paging in/out from memory, it is necessary to consider how the computer can decide which page(s) to replace to allow the requested page to be loaded. When a new page is requested but it is not in memory a page fault occurs.

# Page Replacement Algorithms

(1) First In First Out

(2) Optimal Page replacement: looks forward in line to see which frame it can replace in the event of page fault.

(3) Longest resident: A particular page which is present for the longest time is swapped. (Time of entry should be present in Page Table)

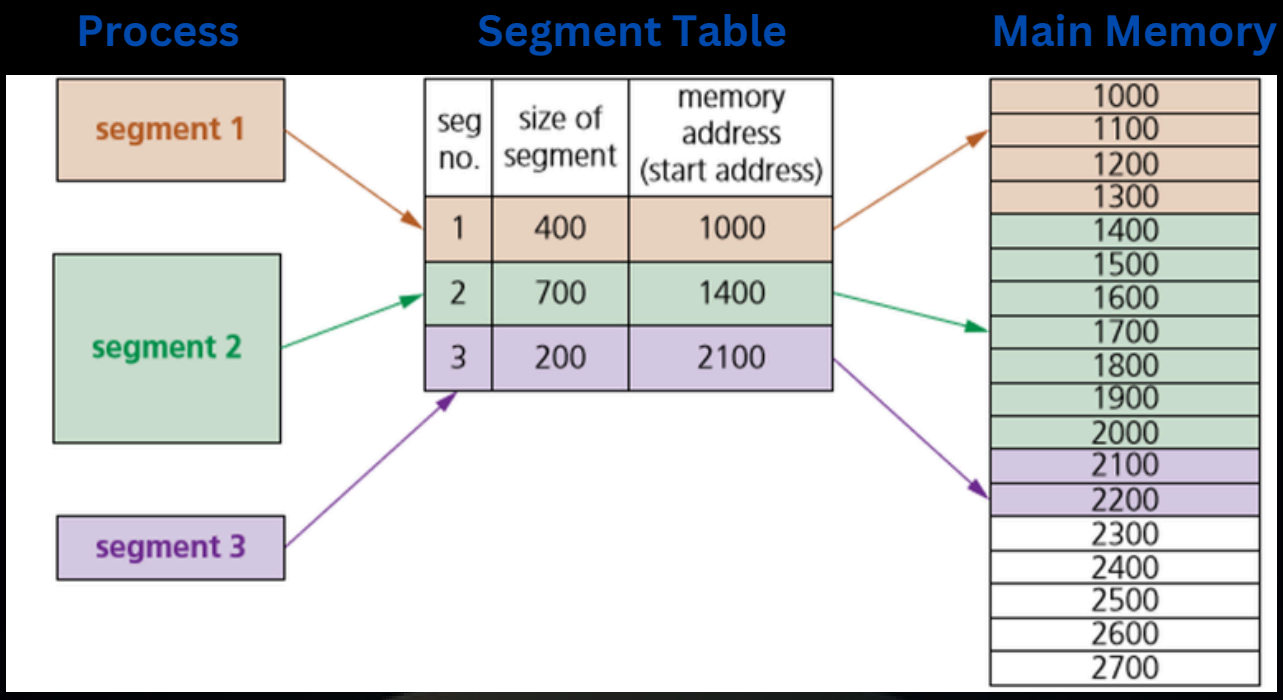
(4) Least used: A particular page which is used less is swapped. (Number of time the page has been accessed should be present in Page Table)



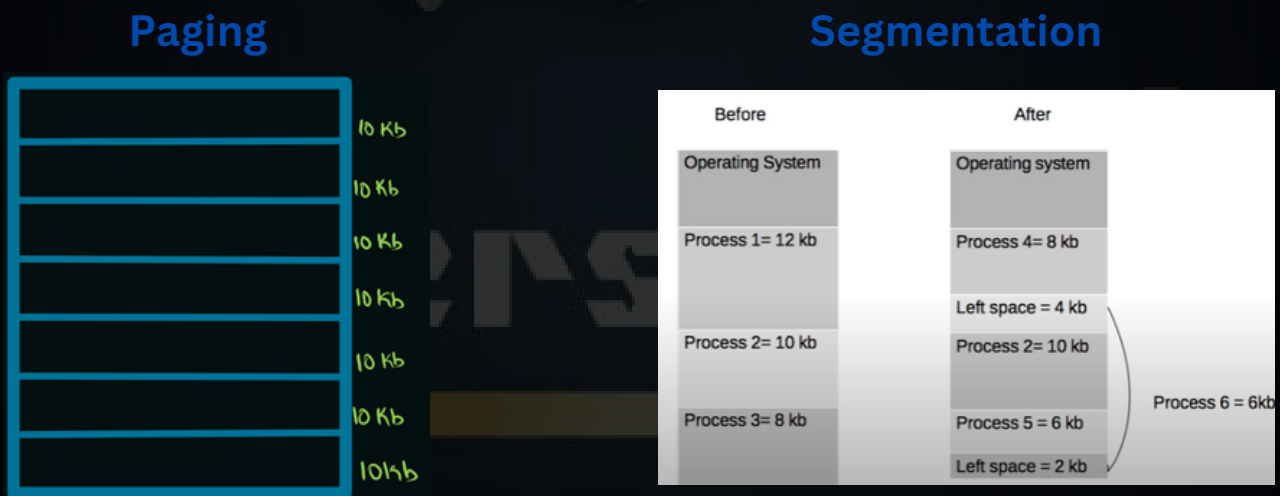
Question : Explain why the algorithms (Longest resident / Least used) may not be the best choice for efficient memory management

- Longest Resident: page in for lengthy period of time may be accessed often ( so not good candidate for being removed. )
- Least Used: A page just entered has a low least value so likely to be a candidate for immediately being swapped out

# Segmentation




# Internal Fragmentation Vs External Fragmentation




- **Internal Fragmentation:** Wasted memory inside allocated blocks because a program doesn't use all the space it was given.
- **External Fragmentation:** Free memory is scattered in small, non-contiguous blocks, making it hard to allocate large chunks of memory.

# Difference Between Paging And Segmentation

| Paging | Segmentation |
| --- | --- |
| A page is a fixed-size block of memory. | A segment is a variable-size block of memory. |
| Since the block size is fixed, it is possible that all blocks may not be fully used. This can lead to internal fragmentation. | Memory blocks are variable-size, this increases the risk of external fragmentation. |
| The operating System divides the memory into pages | The compiler is responsible for calculating the segment size |
| Procedures (Modules) cannot be separated when using paging. | Procedures can be separated when using segmentation. |
| Access Time Faster than segmentation | Access Time slower than paging |

# Virtual Memory


**Question : Describe what is meant by virtual Memory. ?**

- Secondary storage is used to extend ram
- So CPU can access more memory space than available ram
- Only part of program / data in use needs to be in RAM
- Data is swapped between RAM and disk.
- Virtual memory is created temporarily

**Question : Explain how paging is used to manage virtual memory?**

- Divides memory RAM into frames
- Divides virtual memory into blocks of the same size called pages
- Frames/pages are a fixed size
- Sets up a page table to translate logical addresses to physical addresses
- Keeps track of all free frames
- Swaps pages in memory with new pages from disk when needed

**Question : One drawback of using virtual memory is disk thrashing.**
**Describe what is meant by the term disk thrashing?**

- Pages are required back in RAM as soon as they are moved to disk
- There is continuous swapping (of the same pages)
- No useful processing happens
- Because pages that are in RAM and on disk are inter-dependent
- Nearly all processing time is used for swapping pages.

**Question : Explain the circumstances in which disk thrashing could occur?**

- Disk thrashing is a problem that may occur when frequent transfers between main memory and secondary memory take place.
- Disk thrashing is a problem that may occur when virtual memory is being used.
- As main memory fills up, more pages need to be swapped in and out of virtual memory.
- This swapping leads to a very high rate of hard disk head movements.
- Eventually, more time is spent swapping the pages than processing the data.

# Purpose Of An Operating System

## Question 1

8 **(a)** A computer process can be in one of three states.

Identify **and** describe **two** of these states.

State 1 .....

Description .....

.....

.....

State 2 .....

Description .....

.....

.....

[6]

**(b)** One of the main tasks of an operating system is resource management.

Describe how an operating system can maximise the use of resources.

Primary memory .....

.....

.....

.....

.....

.....

Disk .....

.....

.....

.....

.....

.....

[6]

## Question 2

6 **(a)** An operating system (OS) uses a memory management technique called paging.

Explain what is meant by the following terms.

Page .....

.....

.....

Page frame .....

.....

.....

Page table .....

.....

.....

[3]

**(b)** Explain why an operating system needs to use scheduling algorithms.

.....

.....

.....

.....

.....

..... [3]

**(c)** State what is meant by an **interrupt**.

.....

..... [1]

**(d)** For a computer system using multi-programming, the low-level scheduler decides which process will get next use of the processor.

One algorithm could be a round-robin, which means every process gets use of the processor in sequence for a fixed amount of time (time-slice).

For a round-robin algorithm, five processes are currently loaded and get the use of the processor in the sequence:

JOB21 – JOBSS – JOBPT – JOB32 – JOB42, then return to JOB21

Process JOB32 has just completed its time-slice.

The following paragraph describes what happens next. Complete the paragraph by inserting the missing processes.

Interrupt received from the low-level scheduler. Save all register contents for

..... .

Copy the saved registers for ..... to the CPU.

The processor will now process ..... .

[3]

# Question 3

**4** Physical memory is managed using virtual memory and paging.

**(a)** Describe what is meant by **virtual memory**.

.....

.....

.....

..... [2]

**(b) (i)** Explain how paging is used to manage virtual memory.

.....

.....

.....

.....

.....

.....

.....

..... [4]

**(ii)** Give a suitable page replacement algorithm for this process.

..... [1]

**(iii)** One drawback of using virtual memory is disk thrashing.

Describe what is meant by the term **disk thrashing**.

.....

.....

.....

..... [2]

## Question 4

**5** A computer process can be in one of three states: running, ready or blocked.

**(a)** Explain how the processes are affected when the following events take place.

**(i)** The running process needs to read a file from a disk.

.....

.....

.....

.....[2]

**(ii)** The running process uses up its time slice.

.....

.....

.....

.....[2]

**(b)** **(i)** State the conditions that are necessary for a process to move from the ready to the running state.

.....

.....

.....

.....[2]

**(ii)** State the conditions that are necessary for a process to move from the blocked to the ready state.

.....

.....

.....

.....[2]

**(c)** Give **three** reasons why process scheduling is needed.

1 .....

.....

2 .....

.....

3 .....

.....

[3]

# Question 5

**3** A computer operating system (OS) uses paging for memory management.

In paging:

- main memory is divided into equal-size blocks, called page frames
- each process that is executed is divided into blocks of the same size, called pages
- each process has a page table that is used to manage the pages of this process

The following table is the incomplete page table for a process X.

| Page | Presence flag | Page frame address | Additional data |
|---|---|---|---|
| 1 | 1 | 132 | |
| 2 | 1 | 245 | |
| 3 | 1 | 232 | |
| 4 | 0 | 0 | |
| 5 | 1 | 542 | |
| 6 | 0 | 0 | |
| | | | |
| 135 | 0 | 0 | |

When a particular page of the process is currently in main memory, the Presence flag entry in the page table is set to 1.

If the page is not currently present in memory, the Presence flag is set to 0.

**(a)** The page frame address entry for Page 2 is 245.

State what the value 245 could represent.

..... [1]

**(b)** Process X executes until the next instruction is the first instruction in Page 4. Page 4 is not currently in main memory.

State a hardware device that could be storing this page.

..... [1]

**(c)** When an instruction to be accessed is not present in main memory, its page must be loaded into a page frame. If all page frames are currently in use, the contents of a page frame will be overwritten with this new page.

The page that is to be replaced is determined by a page replacement algorithm.

One possible algorithm is to replace the page that has been resident in main memory for the longest time.

**(i)** Give the additional data that would need to be stored in the page table.

.....

..... [1]

**(ii)** Complete the table entries below to show what happens when Page 4 is swapped into main memory. Assume that Page 5 is the one to be replaced.

In the final column, give an example of the data you have identified in **part (c)(i)**.

| Page | Presence flag | Page frame address | Additional data |
|---|---|---|---|
| | | | |
| 4 | ..... | ..... | ..... |
| | | | |

[3]

An alternative algorithm is to replace the page that has been used least.

**(iii)** Give the different additional data that the page table would now need to store.

.....

..... [1]

**(iv)** In the following table, complete the missing data to show what happens when Page 3 is swapped into main memory. Assume that Page 1 is the one to be replaced.

In the final column, give an example of the data you have identified in **part (c)(iii)**.

| Page | Presence flag | Page frame address | Additional data |
|---|---|---|---|
| | | | |
| 3 | ..... | ..... | ..... |
| | | | |

[3]

**(d)** Explain why the algorithms given in **part (c)** may not be the best choice for efficient memory management.

Longest resident .....

.....

.....

.....

Least used .....

.....

.....

..... [4]

## Question 6

**6** A number of processes are being executed in a computer.

**(a)** Explain the difference between a program and a process.

.....

.....

.....

.....[2]

A process can be in one of three states: running, ready or blocked.

**(b)** For each of the following, the process is moved from the first state to the second state. Describe the conditions that cause each of the following changes of the state of a process:

From running to ready .....

.....

.....

.....

From ready to running .....

.....

.....

.....

From running to blocked .....

.....

.....

.....[6]

**(c)** Explain why a process cannot be moved from the blocked state to the running state.

.....

.....

.....

.....

.....

.....[3]

**(d)** Explain the role of the high-level scheduler in a multiprogramming operating system.

.....

.....

.....

.....[2]

# Question 7

6 A number of processes are being executed in a computer.

A process can be in one of three states: running, ready or blocked.

**(a)** For each of the following, the process is moved from the first state to the second state. Describe the conditions that cause each of the following changes of state of a process:

From blocked to ready .....

.....

.....

.....

From running to ready .....

.....

.....

.....[4]

**(b)** Explain why a process cannot move directly from the ready state to the blocked state.

.....

.....

.....

.....

.....

.....[3]

**(c)** A process in the running state can change its state to something which is neither the ready state nor the blocked state.

**(i)** Name this state.

.....[1]

**(ii)** Identify when a process would enter this state.

.....[1]

**(d)** Explain the role of the low-level scheduler in a multiprogramming operating system.

.....

.....

.....

.....[2]

## Question 8

3 **(a)** Draw **one** line to connect each **Operating System (OS)** term to the **most appropriate** description about it.

| OS term | Description |
| --- | --- |
| Multi-tasking | Using secondary storage to simulate additional main memory |
| Paging | Managing the processes running on the CPU |
| Interrupt handling | Managing the execution of many programs that appear to run at the same time |
| Scheduling | Locating non-contiguous blocks of data and relocating them |
| Virtual memory | Transferring control to another routine when a service is required |
| | Reading/writing same-size blocks of data from/to secondary storage when required |

[5]

**(b)** Explain how an interpreter executes a program without producing a complete translated version of it.

.....

.....

.....

.....

.....

.....

.....

..... [4]

## Question 9

**8** Virtual memory, paging and segmentation are used in memory management.

**(a)** Explain what is meant by **virtual memory**.

.....

.....

.....

.....

.....

..... [3]

**(b)** State **one** difference between paging and segmentation in the way memory is divided.

.....

..... [1]

## Question 10

9 **(a)** Explain the need for scheduling in process management.

.....

.....

.....

.....

.....

..... [3]

**(b)** Describe these scheduling routines **and** identify a benefit for each one.

Shortest job first .....

.....

.....

.....

Round robin .....

.....

.....

.....

First come first served .....

.....

.....

.....

[6]

# Question 11

7 **(a)** A student buys a new computer.

State **one** benefit to the student of a user interface **and** give an example.

Benefit .....

.....

.....

Example .....

.....

[2]

**(b)** Two of the process states are the running state and the ready state.

Identify **one other** process state.

..... [1]

**(c)** Outline conditions under which a process could change from the running state to the ready state.

.....

.....

.....

..... [2]

## Question 12

5 **(a)** Outline the reasons why an operating system may need to use virtual memory.

.....

.....

.....

..... [2]

**(b)** Explain the circumstances in which disk thrashing could occur.

.....

.....

.....

.....

.....

..... [3]

# Answers

## Answer 1

| | | |
|---|---|---|
| 8(a) | **1 mark** per bullet point<br>**1 mark** for identifying the state, **max 2** for description<br>M**ax 3** marks for each state<br><br>∞ Ready<br>∞ The process is not being executed<br>∞ The process is in the queue…<br>∞ … waiting for the processor's attention / time slice<br><br>∞ Running<br>∞ The process is being executed by the processor<br>∞ The process is currently using its allocated processor time / time slice<br><br>∞ Blocked<br>∞ The process is waiting for an event …<br>∞ … so it cannot be executed at the moment<br>∞ …e.g. input/output | **6** |
| 8(b) | For up to 2 maximisation techniques for each of memory and disk<br>Max 2 for Memory, Max 2 for disk if no descriptions are given<br><br>**1 mark** for identification of maximisation technique**, 1 mark** for description, **1 mark** for further description or information about improvement to **max 4 for memory**<br><br>**Memory**<br>∞ Moving frequently accessed instructions to cache (1) for faster recall (1) as SRAM is used rather than DRAM for cache (1)<br>∞ Making use of virtual memory (1) with paging or segmentation (1) to swap memory to and from a disk (1)<br>∞ Partitioning memory (1) dividing main memory into static/dynamic partitions (1) to allow for more than one program/task to be available //multiprogramming (1)<br>∞ Removing unused items/tasks from RAM (1) by marking a partition as available (1) as soon as the process using it has terminated (1)<br><br>**1 mark** for identification of maximisation technique, **1 mark** for description, **1 mark** for further description or information about improvement to **max 4 for disk**<br><br>**Disk**<br>∞ Disk caching (1) a disk cache holds data that is frequently transferred to/from the disk (1) the cache can be held on disk or in RAM (1)<br>∞ Compression utility (1) decreasing the size of a file stored on disk (1) in order fit more / larger files on the disk (1)<br>∞ Defragmentation utility (1) files are rearranged to occupy contiguous disk space (1) this reduces the time taken to access files// decreases latency (1) | **6** |

## Answer 2

| | | |
|---|---|---|
| 6(a) | **1 mark** per bullet point<br><br>Page:<br>• Virtual Memory is divided into blocks of a fixed size<br><br>Page frame:<br>• the main memory is divided into page frames of the same size as a page<br><br>Page table:<br>• the Page (Map) table shows the mapping of pages to page frames | **3** |
| 6(b) | **1 mark** per bullet point to **max 3**<br><br>• To allow multiprogramming / multitasking to take place<br>• To ensure fair usage of the processor<br>• To ensure fair usage of peripherals<br>• To ensure fair usage of memory<br>• To ensure higher priority tasks are executed sooner<br>• To ensure all processes have the opportunity to finish | **3** |
| 6(c) | A **signal** from a software **source** or hardware device seeking the **attention of the processer** | **1** |
| 6(d) | **1 mark** per bullet point in the order given<br><br>• JOB32<br>• JOB42<br>• JOB42 | **3** |

## Answer 3

| | | |
|---|---|---|
| 4(a) | 1 mark per bullet point to **max 2**<br><br>• Disk / secondary storage is used to extend the RAM / memory available<br>• … so CPU can access more memory space than available RAM<br>• Only part of program / data in use needs to be in RAM<br>• Data is swapped between RAM and disk | **2** |
| 4(b)(i) | 1 mark per bullet point to **max 4**<br><br>• Divide memory / RAM into frames<br>• Divide virtual memory into blocks of same size called pages<br>• Frames / pages are a fixed size<br>• Set up a page table to translate logical to physical addresses<br>• Keep track of all free frames<br>• Swap pages in memory with new pages from disk when needed | **4** |
| 4(b)(ii) | First-in-first-out // least-recently-used page // least-used-page | **1** |
| 4(b)(iii) | 1 mark per bullet point to **max 2**<br><br>• Pages are required back in RAM as soon as they are moved to disk<br>• There is **continuous** swapping (of the same pages)<br>• No useful processing happens // deadlock<br>• … (because) pages that are in RAM and on disk are inter-dependent<br>• … (nearly) all processing time is used for swapping pages | **2** |

## Answer 4

| | | |
|---|---|---|
| 5(a)(i) | 1 mark per bullet point:<br><br>∞ Running process is halted<br>∞ Process moves to blocked state | **2** |
| 5(a)(ii) | 1 mark per bullet point max 2:<br><br>∞ Running process is halted // another process has use of the processor<br>∞ Process moves to ready state<br>∞ … Until next time slice allocated | **2** |
| 5(b)(i) | 1 mark per bullet point:<br><br>∞ Current process no longer running // processor is available<br>∞ Process was at the head of the ready queue / / process has highest priority | **2** |
| 5(b)(ii) | 1 mark per bullet point:<br><br>∞ The only<br>∞ Required resource becomes available // event is complete | **2** |
| 5(c) | 1 mark per bullet point to max 3:<br><br>∞ to allow multiprogramming<br>∞ to give each process a fair share of the CPU time<br>∞ to allow all processes to complete in a reasonable amount of time<br>∞ to allow highest priority jobs to be executed first<br>∞ to keep the CPU busy all the time<br>∞ to service the largest possible number of jobs in a given amount of time<br>∞ to minimize the amount of time users must wait for their results<br>∞ to maximise the use of peripherals | **3** |

## Answer 5

**3** **(a)** The 245th page frame from the start of memory
// the 245th page frame from some base address [1]

**(b)** Flash memory // magnetic disk // hard drive [1]

**(c)** **(i)** Time of entry (NOT time in memory) [1]

**(ii)**

| Page | Presence Flag | Page frame address | Additional data |
|---|---|---|---|
| 4 | 1 | 542 | 12:07:34:49 |

[1 +1 + 1]

**(iii)** Number of times the page has been accessed [1]

**(iv)**

| Page | Presence Flag | Page frame address | Additional data |
|---|---|---|---|
| 3 | 1 | 132 | 0 |

[1 +1 + 1]

Accept only zero for 'additional data'

**(d)** For example:
**Longest resident:** page in for lengthy period of time may be being accessed often [1]
… so not a good candidate for being removed [1]

**Least used**: a page just entered has a low least used value … [1]
so likely to be a candidate for immediately being swapped out [1]

## Answer 6

**6** **(a)** A program is the written code ("static") [1]
A process is the executing code ("dynamic") [1]

**(b)** **running, ready:**
when process is executing it is allocated a time slice (running state) // process is allocated time on processor [1]
when time slice completed process / interrupt occurs can no longer use processor even though it is capable of further processing (ready state) [1]

**ready, running:**
process is capable of using processor (ready state) [1]
OS allocates processor to process so that process can execute (running state) [1]

**running, blocked:**
process is executing (running state) when it needs to perform I / O operation [1]
placed in blocked state – until I / O operation completed [1]

**(c)** when I / O operation completed for process in blocked state [1]
process put in ready state [1]
OS decides which process to allocate to processor from the ready queue [1]

**(d)** **high-level scheduler:**
decides which processes are to be loaded from backing store [1]
into memory / ready queue [1]

# Answer 7

6 **(a) blocked → ready:**
process is waiting for resource/ I/O operation to complete (blocked state) [1]
when I/O operation completed process goes into ready queue (ready state) [1]
**running → ready:**
when process is executing it is allocated a time slice (running state) // process is allocated time on processor [1]
when time slice completed/interrupt occurs process can no longer use processor even though it is capable of further processing (ready state) [1]

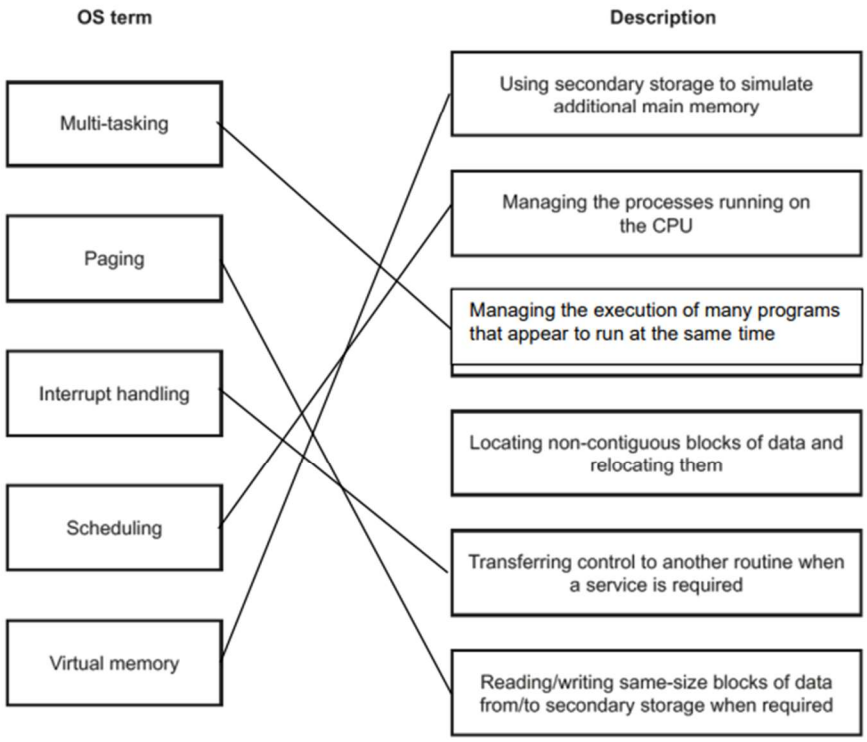
**(b)** to be in blocked state process must initiate some I/O operation [1]
to initiate operation process must be executing [1]
if process in ready state cannot be executing/must be in running state [1]

**(c) (i)** exit/termination/completion [1]

**(ii)** when the process has finished execution [1]

**(d) low-level scheduler:**
decides which of the processes in ready state [1]
should get use of processor/be put in running state [1]
based on position/priority [1]
invoked after interrupt/OS call [1]
[max. 2]

## Answer 8

| Question | Answer | Marks |
|---|---|---|
| 3(a) | **One** mark for each correct line from Operating System Term to Description<br>  | **5** |
| 3(b) | **One** mark for each correct statement **(Max 4)**<br>• An interpreter examines source code one statement at a time<br>• Check each statement for errors<br>• …If no error is found the statement is executed<br>• …If an error is found this is reported and the interpreter halts<br>• Interpretation is repeated for every iteration in repeated sections of code/in loops<br>• Interpretation has to be repeated every time the program is run | **4** |

# Answer 9

| Question | Answer | Marks |
|---|---|---|
| 8(a) | **One** mark for each correct point (**Max 3**)<br><br>• Disk / secondary storage is used to extend the RAM / memory available<br>• … so the CPU appears to be able to access more memory space than the available RAM<br>• Only the data in use needs to be in main memory so data can be swapped between RAM and virtual memory as necessary<br>• Virtual memory is created temporarily. | 3 |
| 8(b) | **One** mark for a correct statement about the difference between paging and segmentation e.g.<br><br>• Paging allows the memory to be divided into fixed size blocks and Segmentation divides the memory into variable sized blocks.<br>• The operating system divides the memory into pages, the compiler is responsible for calculating the segment size.<br>• Access times for paging is faster than for segmentation. | 1 |

# Answer 10

| Question | Answer | Marks |
|---|---|---|
| 9(a) | **One** mark for each point (**Max 3**)<br><br>• Process scheduling allows more than one program/task to appear to be executed at the same time / enables multi-tasking / multiprogramming.<br>• To allow high priority jobs to be completed first.<br>• To keep the CPU busy all the time<br>• … to ensure that all processes execute efficiently<br>• … and to have reduced wait times for all processes / to ensure all processes have fair access to the CPU / prevent starvation of some processes. | 3 |

| Question | Answer | Marks |
|---|---|---|
| 9(b) | **One** mark for each point (**Max 2**) for **Shortest job first**:<br><br>• Process are executed in ascending order of the amount of CPU time required // Short processes are executed first **and** followed by longer processes.<br>• …which leads to an increased throughput (because more processes can be executed in a smaller amount of time).<br><br>**One** mark for each point (**Max 2**) for **Round robin**:<br><br>• Each process is served by the CPU for a fixed time/time slice (so all processes are given the same priority).<br>• Starvation doesn't occur (because for each round robin cycle, every process is given a fixed time/time slice to execute).<br><br>**One** mark for each point (**Max 2**) for **First come first served**:<br><br>• No complex logic, each process request is queued as it is received and executed one by one.<br>• Starvation doesn't occur (because every process will eventually get a chance to run) // less processor overhead. | 6 |

## Answer 11

| Question | Answer | Marks |
|---|---|---|
| 7(a) | **One** mark for a benefit (**Max 1**) e.g.<br>MP1 The user interface hides the complexities of the computer hardware/operating system from the user<br>MP2 It provides appropriate access systems for users with differing needs<br>MP3 Complex commands involving memory locations/buses/computer hardware/ are avoided<br><br>**One** mark for a valid example (**Max 1**) e.g.<br>Clicking on icon rather than writing code<br>Using a graphical user interface / icons for navigation | **2** |
| 7(b) | Blocked (state) | **1** |
| 7(c) | **One** mark per mark point (**Max 2**)<br>MP1 When the time slice of the running process expires (round robin).<br>MP2 …and there is a process with a higher priority in the ready queue, the running process is pre-empted<br>MP3 When an interrupt arrives at the CPU, (the process running on the CPU gets pre-empted). | **2** |

## Answer 12

| Question | Answer | Marks |
|---|---|---|
| 5(a) | **One** mark per mark point (**Max 2**)<br>MP1 Virtual memory is used when RAM is running low<br>MP2 …such as when a computer is running many processes at once.<br>MP3 Virtual memory may be used for efficient use of RAM / the processor<br>MP4 …such as if data / programs are not immediately needed, they can be moved from RAM to virtual memory | **2** |
| 5(b) | **One** mark per mark point (**Max 3**)<br>MP1 Disk thrashing is a problem that may occur when frequent transfers between main memory and secondary memory take place // Disk thrashing is a problem that may occur when virtual memory is being used<br>MP2 As main memory fills up, more pages need to be swapped in and out of secondary/virtual memory<br>MP3 This swapping leads to a very high rate of hard disk head movements<br>MP4 Eventually, more time is spent swapping the pages/data than processing the data. | **3** |