OCR Computer Science AS Level

1.2.1 Systems Software

Intermediate Notes

Specification:

1.2.1 a)

- **Operating Systems**
    - The need for, function and purpose of operating systems

1.2.1 b)

- **Memory Management**
    - Paging
    - Segmentation
    - Virtual Memory

1.2.1 c)

- **Interrupts**
    - Role of interrupts
    - Role of ISR (Interrupt Service Routines)
    - Role of interrupts within Fetch-Decode- Execute Cycle

1.2.1 d)

- **Scheduling**
    - Round robin
    - First come first served
    - Multi-level feedback queues
    - Shortest job first
    - Shortest remaining time

1.2.1 e)

- **Types of operating system**
    - Distributed
    - Embedded
    - Multi-tasking
    - Multi-user
    - Real Time

1.2.1 f)

- **BIOS**

1.2.1 g)

- **Device drivers**

1.2.1 h)

- **Virtual machines**
    - Where software is used to take on the function of a machine for:
        - Executing intermediate code
        - Running an operating system within another

Operating Systems

The term ‘operating system’ refers to a collection of programs that provide an interface between the user and computer. Operating systems enable the user to communicate with the computer and perform tasks involving the management of computer memory and resources.
Popular desktop operating systems: Windows, macOS
Popular mobile phone operating systems: iOS, Android.

Operating systems provide the following features:

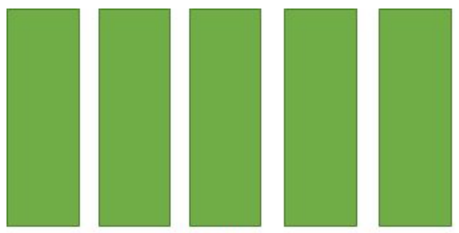
- Memory management
- Resource management eg. scheduling
- File management
- Input/ Output management
- Interrupt management
- Utility software
- Security
- User interface

Memory Management

Computer memory must be shared fairly between multiple programs and applications being used. Often, main memory is not large enough to store all of the programs being used.
Paging, segmentation and virtual memory are techniques used by the operating system to ensure memory is shared effectively by programs.

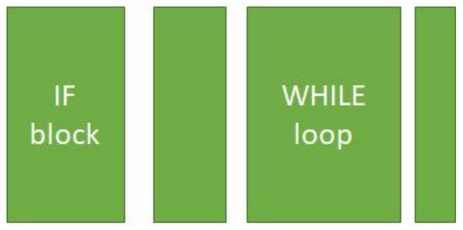
Paging

Paging is when memory is split up into equal-sized sections known as pages.. These can then be swapped between main memory and the hard disk as needed.


Paging

Segmentation

Segmentation is the splitting up of memory into logical sized divisions, known as segments, which vary in size. These represent the structure and logical flow of the program.



Segmentation

Virtual Memory

Virtual memory uses a section of the hard drive to act as RAM when the space in main memory is insufficient to store programs being used. Sections of programs not currently being used are temporarily moved into virtual memory through paging, freeing up memory for other programs in RAM.

The key issue with using these techniques is disk thrashing. This is when the computer ‘freezes’ due to pages being swapped too frequently between the hard disk and main memory. This issue becomes progressively worse as virtual memory is filled up.

Interrupts

Interrupts are signals generated by software or hardware to indicate to the processor that a process needs attention. Interrupts have different priorities and how urgent they are must be taken into account by the operating system when allocating processor time. They are stored within an abstract data structure called a priority queue in an interrupt register. Examples of interrupts include a printer signalling the completion of a print job or a peripheral signalling power failure.

Interrupt Service Routine

The processor checks the contents of the interrupt register at the end of each Fetch-Decode-Execute cycle. If an interrupt exists that is of a higher priority to the process being executed, the current contents of the special purpose registers in the CPU are temporarily transferred into a stack. The processor then responds to the interrupt by loading the appropriate interrupt service routine (ISR) into RAM. A flag is set to signal the ISR has begun.

Synoptic Link

Abstract **data structures** such as **queues** and **stacks** are discussed further in 1.4.2 Data Structures. A queue is a **FIFO (First In First Out)** data structure in which the first element in the list is the first to leave. A stack is a **LIFO (Last In First Out)** data structure in which the first element in the list is the last to leave.

Once the interrupt has been serviced, the flag is reset. The interrupt queue is checked for further interrupts of a higher priority to the process that was originally being executed. If there are more interrupts to be serviced, the process described above is repeated until all priority interrupts have been serviced. Otherwise, the contents of the stack are transferred back into the registers in memory. The Fetch-Decode-Execute cycle resumes as before.

Synoptic Link

The special purpose memory registers referred to here are those you have encountered in 1.1.1; the PC, MAR, MDR, CIR and ALU!

Scheduling

The operating system ensures all sections of programs being run (known as 'jobs') receive a fair amount of processing time. This is done through various scheduling algorithms, which work in different ways. Scheduling algorithms can either be:

1. Pre-emptive

   Jobs are actively made to start and stop by the operating system.
   For example: Multilevel Feedback Queues, Shortest Remaining Time, Round Robin

2. Non pre-emptive

   Once a job is started, it is left alone until it is completed.
   For example: First Come First Served, Shortest Job First

Note

The scheduling algorithm used depends on the task at hand. For example, **Shortest Remaining Time** is used where the completion of shorter tasks before other tasks is preferred. Typically however, operating systems such as Windows use **multilevel feedback queues** to prevent **process starvation.**

Below are the scheduling algorithms you need to know:

- Round robin

  Each job is given a section of processor time - a time slice - within which it is allowed to execute. Once each job in the queue has used its first time slice, this process is repeated until a job has been completed, at which point it is removed from the queue.
  Advantages:
  - All jobs will eventually be attended to.

  Disadvantages:
  - Longer jobs will take a much longer time for completion as execution is inefficiently split up into multiple cycles.
  - Round robin does not take into account job priority or urgency

- First come first served

  Jobs are processed in chronological order by which they entered the queue.
  Advantages:
  - Straightforward to implement

  Disadvantages:
  - Does not take into account job priority or urgency

- Multilevel feedback queues

  This makes use of multiple queues, each which is ordered based on a different priority.
  Advantages:

- Takes into consideration different job priorities

Disadvantages:

- Difficult to implement

- Shortest job first

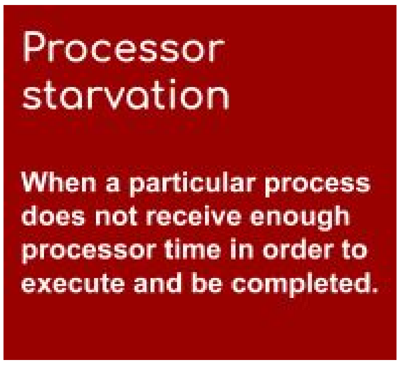
  The queue storing jobs to be processed is ordered according to the time required for completion, with the longest jobs being serviced at the end.

  Advantages:

  - Suited to batch systems, as waiting time is reduced

  Disadvantages:

  - Requires processor to calculate how long each job will take
  - Processor starvation if short jobs are continuously added to the queue
  - Does not take into account job priority or urgency


Processor starvation

When a particular process does not receive enough processor time in order to execute and be completed.

- Shortest remaining time

  The queue storing jobs to be processed is ordered according to the time left for completion, with the jobs with the least time to completion being serviced first.

  Advantages:

  - Throughput is increased as shorter processes can be quickly completed

  Disadvantages:

  - Does not take into account job priority or urgency
  - Processor starvation if short jobs are continuously added to the queue

Types of Operating System

There are various types of operating systems which have different uses.

- Distributed

  Run across multiple devices, allowing the load to be spread across multiple computer processors when a task is run.
- Embedded

  Built to perform a small range of specific tasks and catered towards a specific device such as a household appliance. They are limited in their functionality and hard to update although they consume significantly less power than other types of OS.
- Multi-tasking

  Multi-tasking operating systems enable the user to carry out tasks seemingly simultaneously. This is done by using time slicing to switch quickly between programs and applications in memory.
- Multi-user

  Multiple users make use of one computer. A scheduling algorithm Is used to ensure processor time is shared fairly between jobs and prevent processor starvation.
- Real Time

  Commonly used in time-critical computer systems, a real time OS is designed to perform a task within a guaranteed time frame. Examples of use include the management of control rods at a nuclear power station or within self-driving cars: any situation where a response within a certain time period is crucial to safety.

BIOS

The Basic Input Output System is the first program that runs when a computer system is switched on. The BIOS is responsible for running various key tests before the operating system is loaded into memory, such as:

- POST (Power-on self test) which ensures that all hardware (keyboards, disk drives) are correctly connected and functional
- Checking the CPU clock, memory and processor is operational
- Testing for external memory devices connected to the computer

After these checks are completed, the operating system can be loaded into RAM from the hard disk by the bootstrap/ bootloader.

Device Drivers

Synoptic Link

Device drivers are an example of abstraction as you as a user are able to interact with hardware without knowing details about how this works. This concept is explored further in 2.2.2

Device drivers are computer programs which are provided by the operating system and allow the operating system to interact with hardware.

When a piece of hardware is used, the device driver communicates this request to the operating system which can then produce the relevant output.

Device drivers are specific to the computer's architecture, so different drivers must be used for different device types such as smartphones, games consoles and desktop PCs. Device drivers are also specific to the operating system installed on the device.

Virtual Machines

A virtual machine is a theoretical computer in that it is asoftware implementation of a computer system. It provides an environment with a translator for intermediate code to run.

Synoptic Link

Translators are discussed further in 1.2.2.

Intermediate Code

Code that is halfway between machine code and object code is called intermediate code. This is independent of the processor architecture so can be runacross different machines and operating systems.

Virtual machines are commonly used to test programs on different operating systems, saving the time and money of purchasing multiple devices for testing. However, running intermediate code in a virtual machine can be considerably slower.

Other uses of virtual machines include:

- Protection from malware

  Malware will affect the virtual machine rather than the device being used.
- Running incompatible software

  Programs specific to different operating systems or different versions of an operating system can be run within a VM, saving time and money required to purchase the hardware.