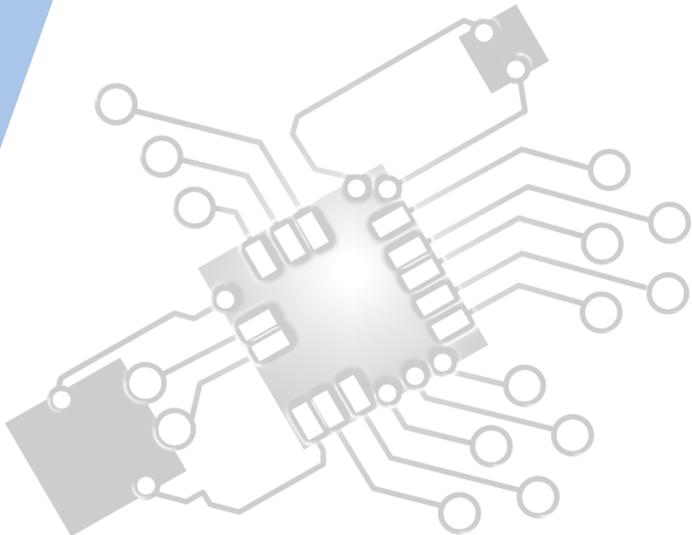




Resource Management

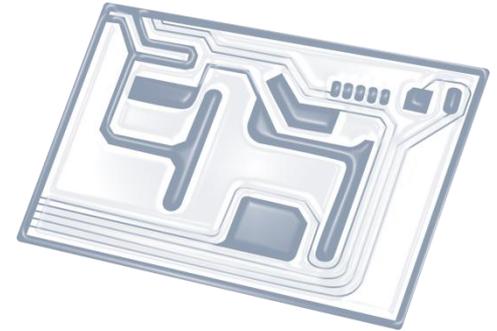
IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department



HL Topics 1-7, D1-4



1: System design



2: Computer Organisation



3: Networks



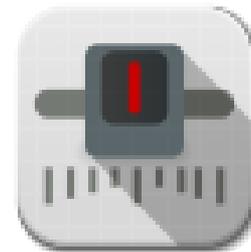
4: Computational thinking



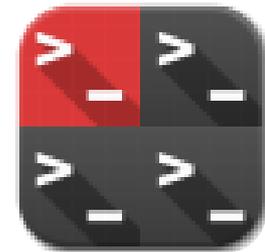
5: Abstract data structures



6: Resource management



7: Control



D: OOP

HL *only* 6 Overview

System resources

6.1.1 Identify the resources that need to be managed within a computer system

6.1.2 Evaluate the resources available in a variety of computer systems

6.1.3 Identify the limitations of a range of resources in a specified computer system

6.1.4 Describe the possible problems resulting from the limitations in the resources in a computer system

Role of the operating system

6.1.5 Explain the role of the operating system in terms of managing memory, peripherals and hardware interfaces

6.1.7 Outline OS resource management techniques: scheduling, policies, multitasking, virtual memory, paging, interrupt, polling

6.1.8 Discuss the advantages of producing a dedicated operating system for a device

6.1.9 Outline how an operating system hides the complexity of the hardware from users and applications



1: System design

2: Computer Organisation



3: Networks

4: Computational thinking



5: Abstract data structures

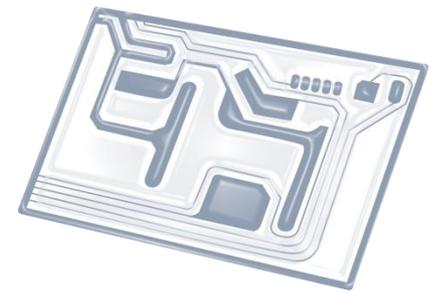
6: Resource management



7: Control

D: OOP



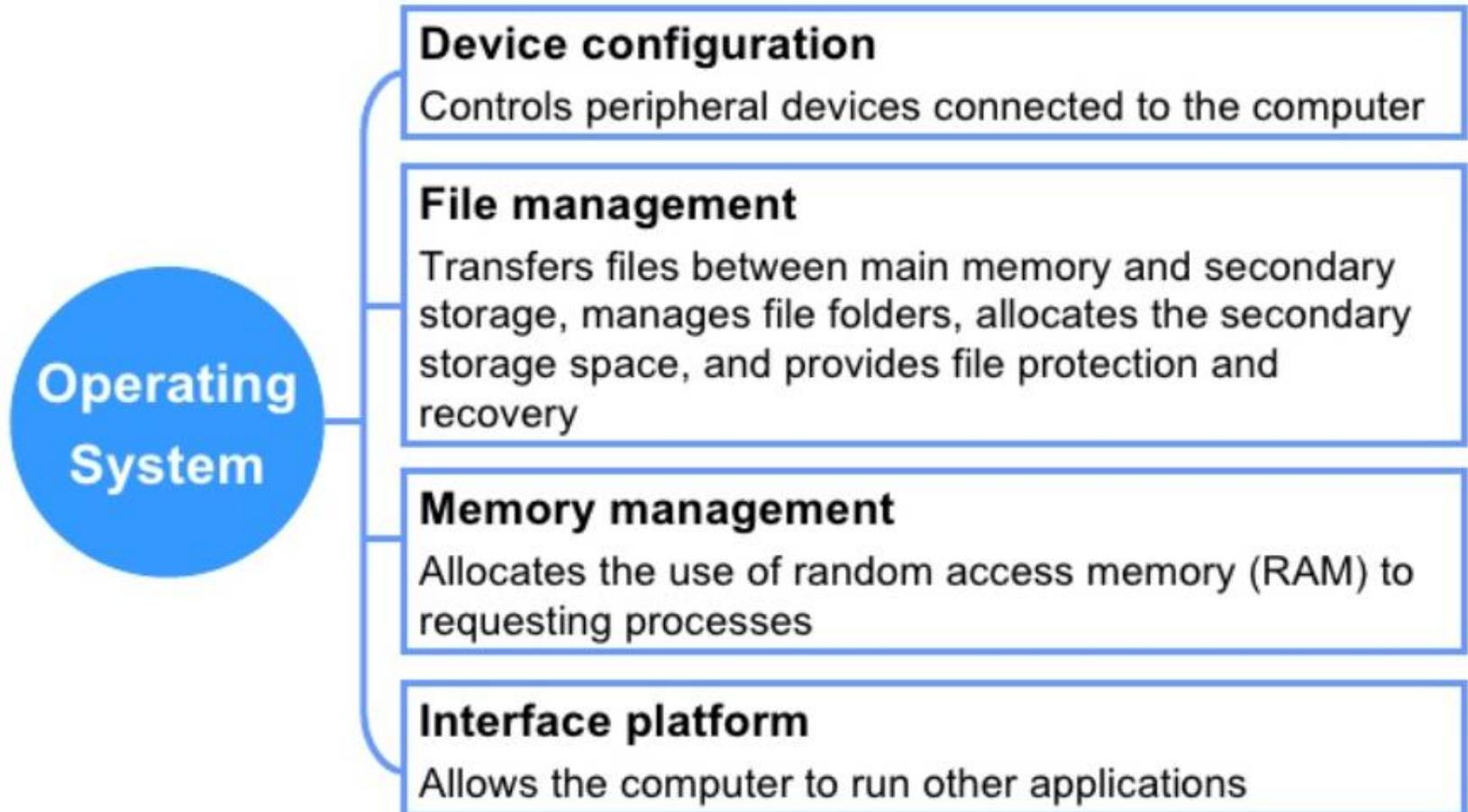


Topic 6.1.5

Explain the role of the **operating system** in terms of managing **memory**, **peripherals** and **hardware interfaces**

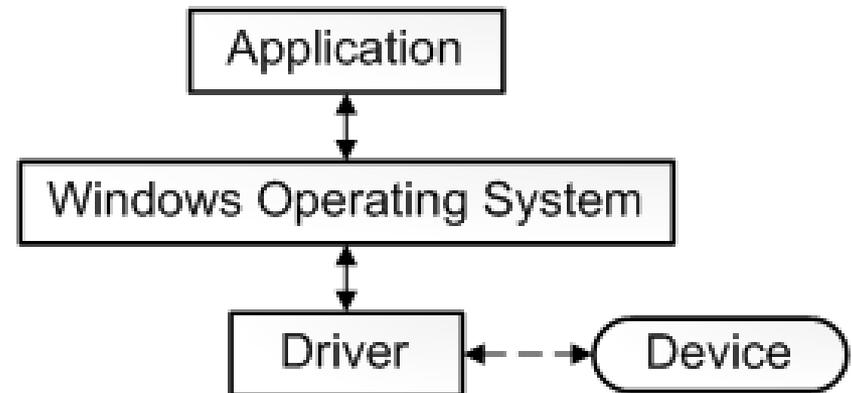
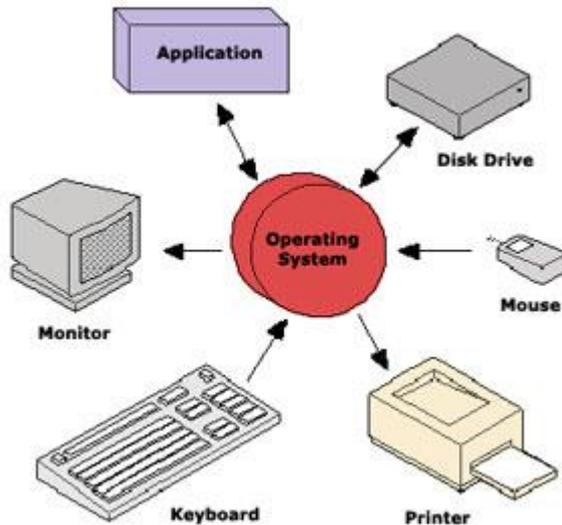


Functions of an operating system



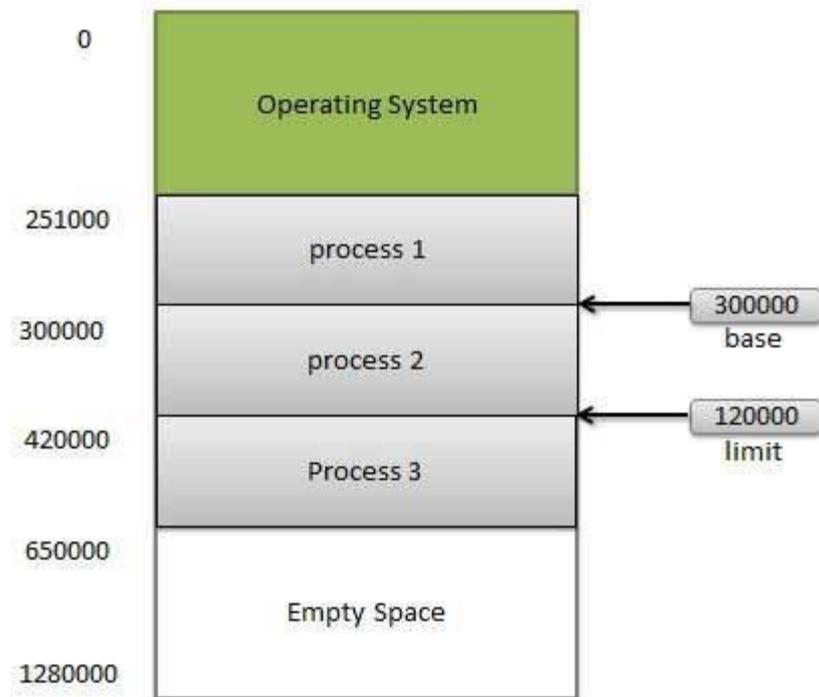
Controls (peripheral) devices

- Through the use of **drivers** (specially written, individualised) *'translation'* programs, the other programs (and ultimately the user) can use and control peripheral devices (like a keyboard, mouse, printer, etc.)



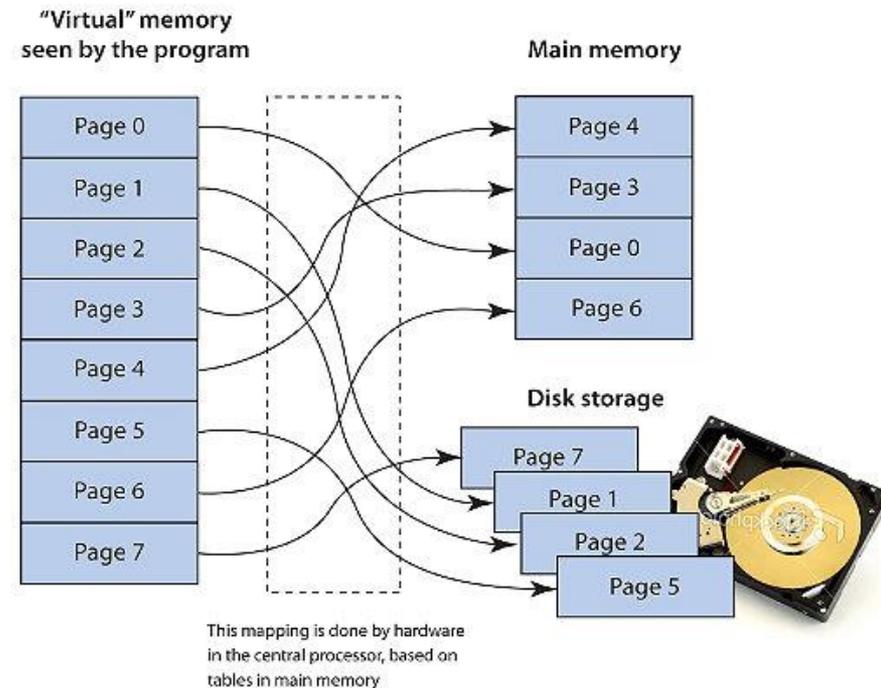
Managing (primary) memory

- The OS has to ensure that each process (program) runs in its own allocated memory space.
- If programs interfere with each other's memory space it could cause many problems including corruption and security issues.



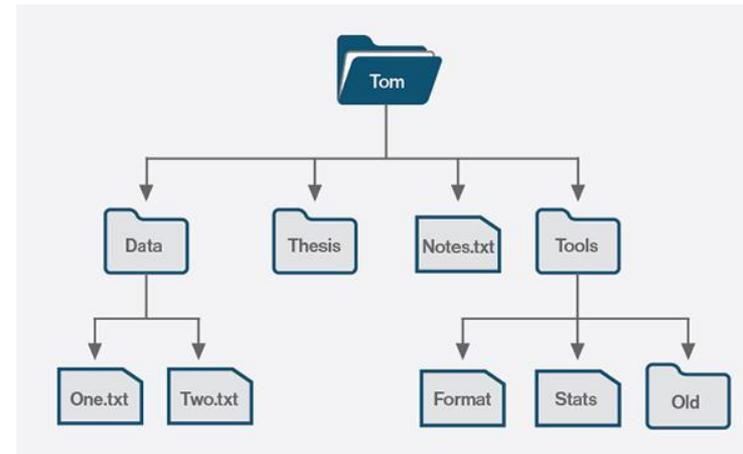
Virtual memory

- **Virtual memory** is a feature of an operating system (OS) that allows a computer to compensate for shortages of physical **memory** by temporarily transferring pages of data from random access **memory** (RAM) to disk storage.



Managing (secondary) storage

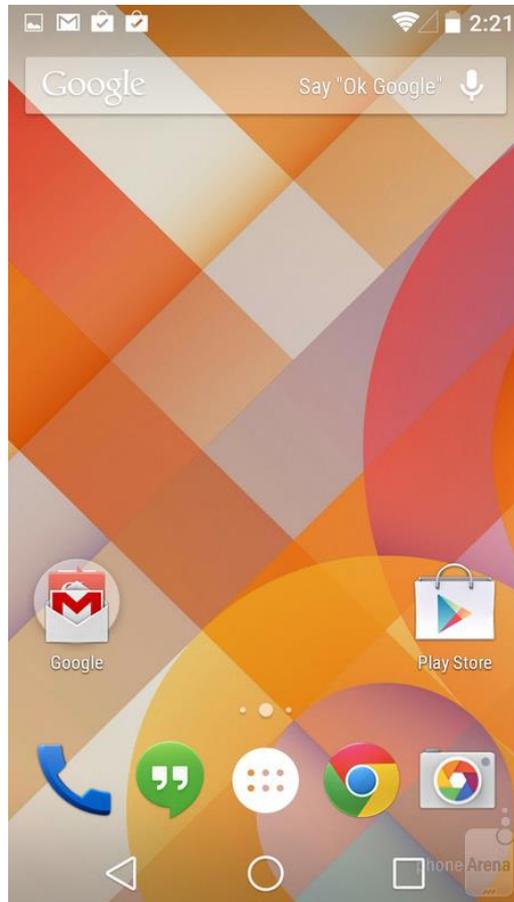
- The OS manages the secondary storage by providing structure and access methods to these structures
- We often refer to this as the **folder-structure** but in some OSES it is referred to as the **directory structure**.
- The OS also manages the security access of these folders



Provides an interface

- *User interface* is used to interact with the computer to perform various tasks. User gives commands to computer and enters the data into computer. The operating system then translates the input/output and sends it to the correct memory address/folder address to be processed.
- **Types of Operating Systems**
Based on the user interface, there are two types of operating systems.
- Graphical User Interface Operating System (Windows)
- Command Line Operating System (Linux Terminal)

Android vs Windows Phone (GUI)



Windows vs OS X (Apple)

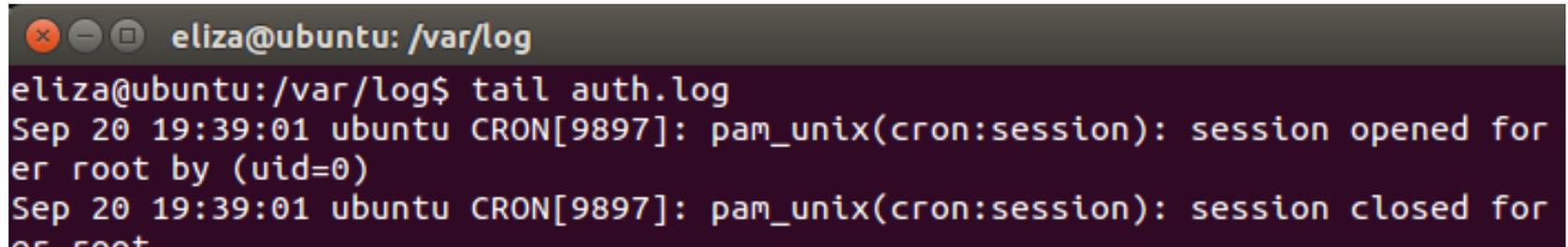


Windows vs Linux vs OSX terminals

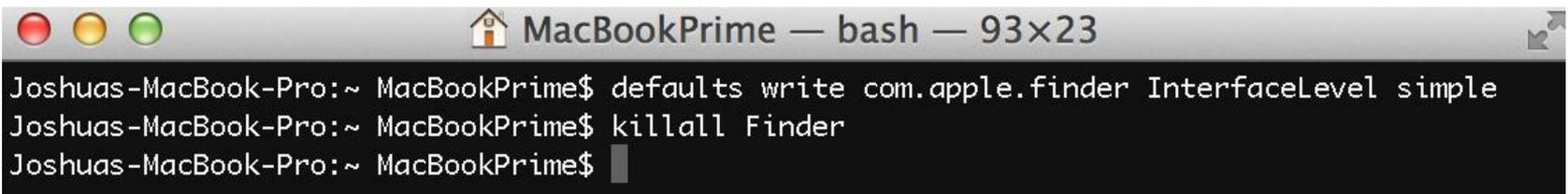
A screenshot of a Windows PowerShell terminal window. The title bar reads 'Windows PowerShell'. The text inside the window shows the copyright notice for Microsoft Corporation and the current directory path.

```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator>
```

A screenshot of a Linux terminal window on an Ubuntu system. The prompt is 'eliza@ubuntu: /var/log'. The user has run the command 'tail auth.log', and the output shows two lines of log entries related to a cron session for root.

```
eliza@ubuntu: /var/log
eliza@ubuntu: /var/log$ tail auth.log
Sep 20 19:39:01 ubuntu CRON[9897]: pam_unix(cron:session): session opened for
er root by (uid=0)
Sep 20 19:39:01 ubuntu CRON[9897]: pam_unix(cron:session): session closed for
er root
```

A screenshot of a MacOS terminal window on a MacBook Prime. The title bar reads 'MacBookPrime — bash — 93x23'. The user has run two commands: 'defaults write com.apple.finder InterfaceLevel simple' and 'killall Finder'.

```
Joshuas-MacBook-Pro:~ MacBookPrime$ defaults write com.apple.finder InterfaceLevel simple
Joshuas-MacBook-Pro:~ MacBookPrime$ killall Finder
Joshuas-MacBook-Pro:~ MacBookPrime$
```

Time-slicing

- With a **multi-user system**, a time-slice is the set amount of processing time each user gets.
- With a **single-user system**, a time-slice is the set amount of processing time each program gets.
- Slices (also called threads) are alternately processed to give the illusion of many tasks happening at once.

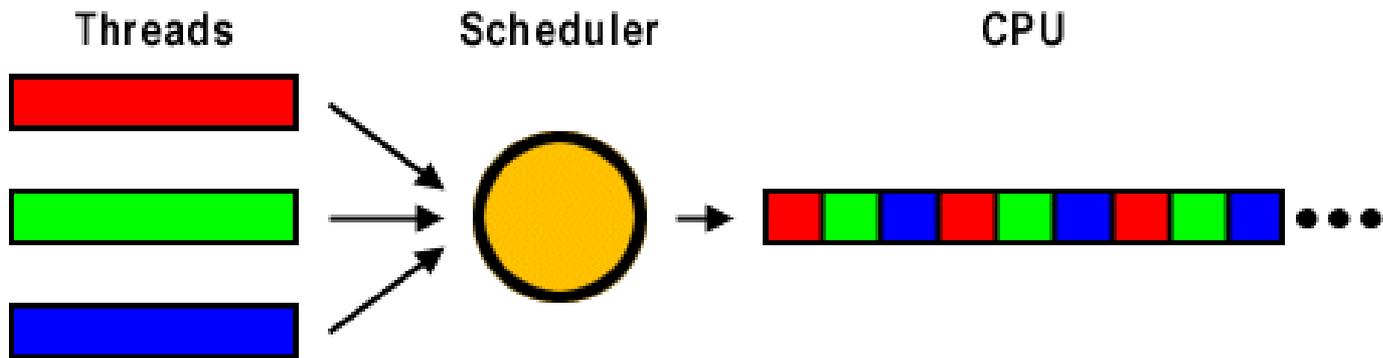
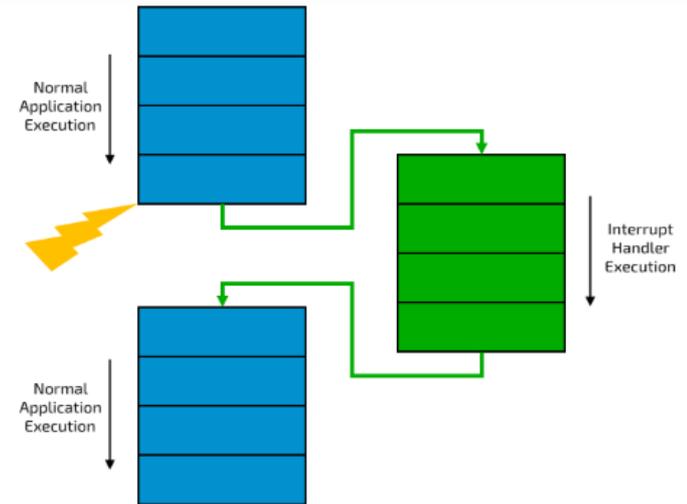
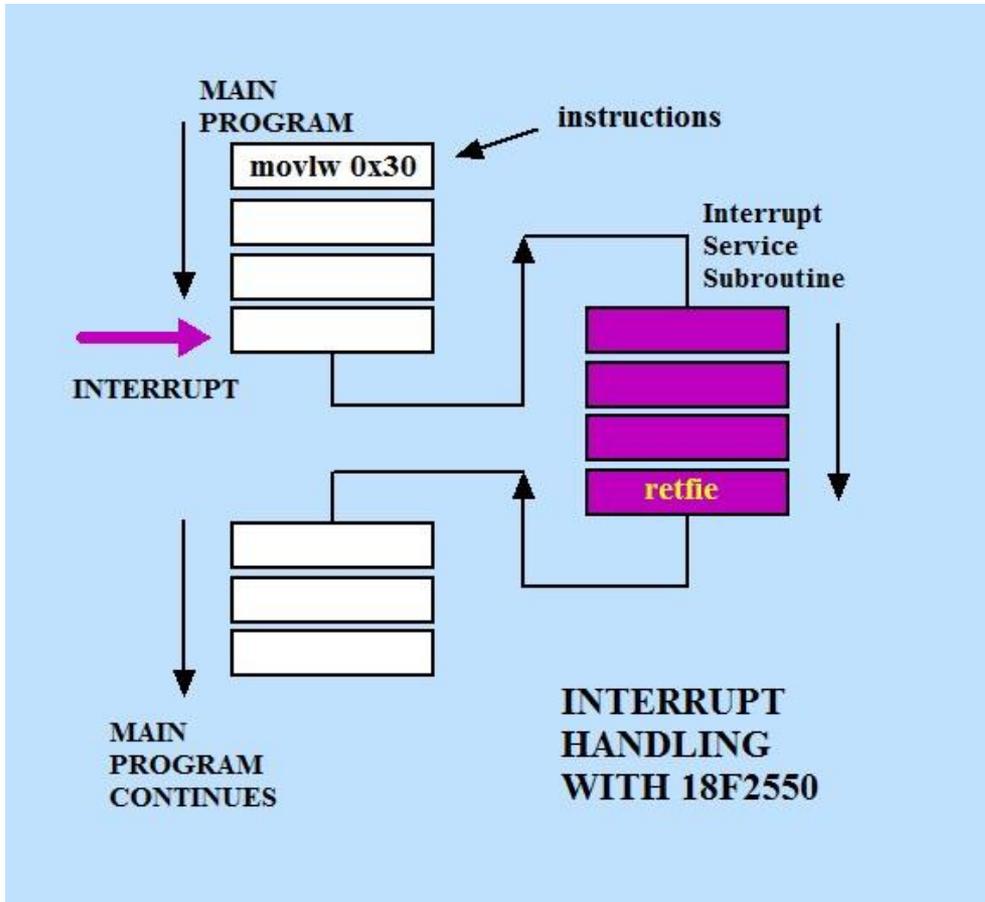


Fig. 1: Thread scheduling

Interrupt handling

- An interrupt handler is a function in of the OS or a device driver, whose execution is triggered by the reception of an interrupt.
- In general, interrupts are used to handle high-priority conditions that require the interruption of the current code the processor is executing.
- For example, pressing a key on a keyboard, or moving the mouse, triggers interrupts that call interrupt handlers which read the key, or the mouse's position, and copy the associated information into the computer's memory.

Waiting for input trigger



Can you discuss the following?

- Allocating storage and keeping track of programs in memory
- Swapping between programs on time-slicing, priority or when one is waiting for input

