TOPIC 3 Computers Knowledge Organiser

Hardware

Memory
- Random Access Memory (RAM)
  - Volatile
- Read Only Memory (ROM)
  - Non-volatile
- Virtual memory
- Flash memory

Types of computer systems.
Advantages / disadvantages of each.
- General-purpose systems
- Dedicated systems
- Embedded systems
- Expert systems

Hardware
- Definition.
  - Input
  - Process
  - Storage
  - Output
- Von Neumann architecture

Secondary storage
- Magnetic hard disk
- Optical disk
- Flash memory
- Cloud
- Considerations for selecting storage:
  - Capacity
  - Speed
  - Portability
  - Durability
  - Reliability

Central processing unit (CPU)
- Arithmetic & logic unit
- Control unit.
- Fetch-execute cycle
  - Fetch
  - De-code
  - Execute
- The boot sequence
- Clock speed
  - Processor speeds (MHz, GHz)
- Cache memory
- Multiple processor cores
  - Advantages / disadvantages.

Software:

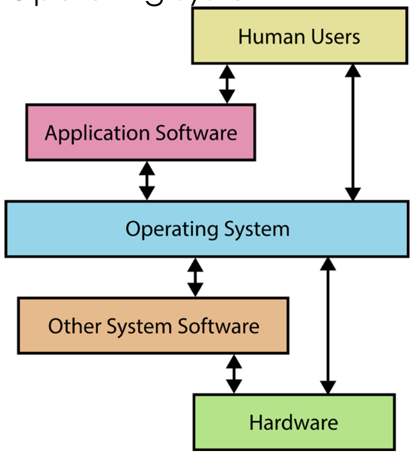
Software

What do they do and what are they used for?

- System Software (Operating System)
- Application
- Utility

System software

- **Software that controls the hardware.**
  - Operating system



Robust software

- Audit trails
- Code reviews

Programming software

- Editors (MU)
- Interpreters
  - Compilers
  - Translators

Functions of the OS.

- File management
- Process management
- Peripheral management
- User management

Programming languages.

- Low level languages:
  - Machine language
  - Assembly language
    - Mnemonics
- High level languages:
  - Source code
  - Assembler
  - Compiler
  - Interpreter

Utility software

- Anti-malware
- Data compression
- Defragmentation
- Backup
- File Repair

Specification:

Subject content	Students should:	
3.1 Hardware	3.1.1	understand the von Neumann stored program concept and the role of main memory (RAM), CPU (control unit, arithmetic logic unit, registers), clock, address bus, data bus, control bus in the fetch-decode-execute cycle
	3.1.2	understand the role of secondary storage and the ways in which data is stored on devices (magnetic, optical, solid state)
	3.1.3	understand the concept of an embedded system and what embedded systems are used for
3.2 Software	3.2.1	understand the purpose and functionality of an operating system (file management, process management, peripheral management, user management)
	3.2.2	understand the purpose and functionality of utility software (file repair, backup, data compression, disk defragmentation, anti-malware)
	3.2.3	understand the importance of developing robust software and methods of identifying vulnerabilities (audit trails, code reviews)
3.3 Programming languages	3.3.1	understand the characteristics and purposes of low-level and high-level programming languages
	3.3.2	understand how an interpreter differs from a compiler in the way it translates high-level code into machine code