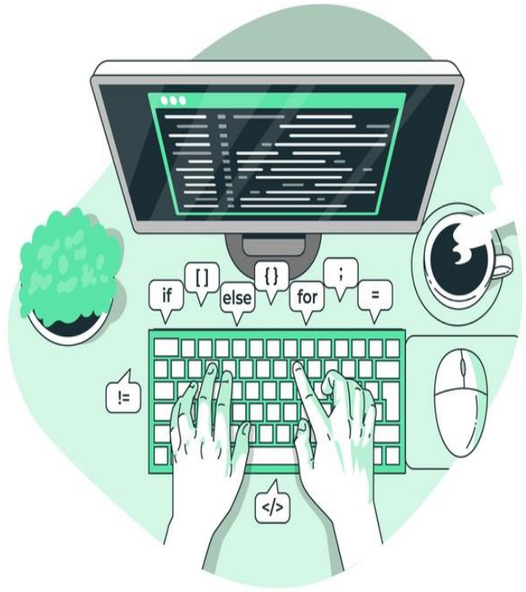
Operating Systems (OS)

- Provide an interface between the user and computer
- Features include Memory management, Resource management, File management, Input Output Management, Interrupt management, Utility software, Security, User interface

Algorithms

- A set of instructions used to solve a set problem.
- Inputs must be clearly defined.
- Must always produce a valid output.
- Must be able to handle invalid inputs.
- Must always reach a stopping condition.
- Must be well-documented for reference.
- Must be well-commented.



Memory Management

- Computers often need more memory than is available and so must efficiently manage the available memory and share it between programs.

Paging

- Memory is broken down into equal sized parts called pages.
- Pages are swapped between main and virtual memory.

Segmentation

- Memory is split up into segments.
- Segments can vary in size.
- These segments represent the logical flow and structure of a program.

Virtual Memory

- Part of the hard drive can be used as RAM.
- Access is slower than RAM.
- Paging is used to move sections which are not in active use into virtual memory.

Interrupts

- A signal generated by hardware or software to tell the processor it needs attention.
- Have different priorities.
- Stored with a priority queue in an interrupt register.

Interrupt Service Routine (ISR)

- At the end of the fetch, decode, execute cycle the interrupt register is checked.
- If there is an interrupt with a higher priority than the current task:
  - The contents of the registers are transferred into a stack .
  - The appropriate (ISR) is loaded into RAM.
  - A flag is set, noting that the ISR has begun.
  - The flag is reset when the ISR has finished.
  - This process repeats until no more interrupts exist.

Virtual Machines

- A software implementation of a virtual computer
- Intermediate code is halfway between machine code and object code.
- It is independent of process architecture allowing it to run across different systems.
- It takes longer to execute
- Virtual machines can be used to help protect from malware, test software, or run software with different versions or OS requirements.

Applications Software

- Used by an end user to perform a specific task.
- e.g. word processor or web browser

Systems software

- Manages computer resources to maintain performance
- e.g. operating system or device driver.

Utility Software

- Has a specific function to maintain OS performance
- e.g. backup or compression software

Unit 1.2 Software and Software Development (Page 1)

Scheduling

- The operating system schedule processor time between running programs.
- These are known as jobs and held in a queue.
- Pre-emptive scheduling routines actively start and stop jobs
- Non pre-emptive routines start jobs then leave them to complete

Round Robin Routine

- Each job is given a time slice of processor time to run in.
- When a job has used up it's time slice it is returns to the start of the queue and receives another.
- This repeats until the job is complete.

First come first served routine

- Jobs are processed in the order they entered the queue

Multilevel feedback queue routine

- Uses multiple queues, each with a different priority

Shortest job first routine

- The queue is ordered by the amount of processor time needed.
- The shortest jobs are completed first.

Shortest time remaining routine

- The queue is ordered based on the time left to completion.
- Jobs with the least time needed to complete are finished first

	Advantages	Disadvantages
Round Robin	All jobs are eventually attended to.	Longer jobs take much longer. Takes no account of priority.
First Come First Served	Easy to implement.	Takes no account of priority.
Multilevel Feedback	Considers job priority.	Tricky to implement
Shortest Job First	Works well for batch systems	Requires additional processor time to order the queue Takes no account of priority.
Shortest Time remaining	Increased throughput	Requires additional processor time to order the queue Takes no account of priority.

Types of Operating System

Distributed

- Runs across several devices
- Spreads task load across multiple computers

Embedded

- Built to perform a specific small task
- Built for a specific device and hardware
- Limited functionality
- Less resource intensive

Multi Tasking

- Allows multiple tasks to be completed simultaneously
- Uses time slicing to switch between applications

Multi User

- Several users can use a single computer
- A scheduling algorithm allocates processor time between jobs

Real Time

- Performs tasks within a guaranteed time frame
- Used in time critical systems.

BIOS

- Basic Input Output System.
- Runs when a computer first turns on.
- Runs tests then loads the main OS into memory.
- Power On Self Test (POST) makes sure all hardware is connected and functional
- Tests the CPU, Memory and external devices.

	Open Source	Closed Source
Advantages	Provided along with the source code. No license required to use.	Needs a license to use. Source code is not available. Protected by Copyright
	Online, free, community support. Many individuals will work on the code meaning it is of high quality. Free.	The company provides support and documentation. Professionally developed. More secure. Regular updates
Disadvantages	Not always well supported or documented. Variable quality code. Less secure.	Code cannot be customised to meet user needs. License may restrict use. More expensive.

Translators

- Covert source code into object code.

Compiler

- Translates code all in one go.
- Compilation process is longer.
- Produces platform specific code.
- Complied code can be run without a translator.

Interpreter

- Translates and executes code line by line.
- Will error if a line contains an error.
- Slower to run than compiled code.
- Code is platform independent.
- Useful for testing.

Assembler

- Assembly code is platform specific, low level code.
- Translates assembly code to machine code.
- 1 line of assembly code = 1 line of machine code.

Stages of Compilation

Lexical Analysis

- Comments and whitespace removed
- Identifiers and keywords replaced with tokens
- Token info stored in a symbol table

Syntax Analysis

- Tokens checked against language rules
- Flags syntax errors
- Abstract Syntax Tree Produced

Code Abstraction

- Machine code produced using Abstract Syntax Tree

Optimisation

- Reduces execution time
- Most time consuming part
- Removes redundant code.

Device Drivers

- Code which allows the OS to interact with hardware
- Specific to the OS and architecture type

Linkers

- Link external modules and libraries used in the code.
- Static linkers copy the library code directly into the file, increasing its size.
- Dynamic linkers just add the addresses of the module or library.

Loaders

- Provided by the OS to fetch the library or module from the given location in memory

Libraries

- Libraries include pre compiled, error free, code which can be used within other programs
- Common functions can quickly and easily be reused across multiple programs
- Saves the time and effort associated with developing and testing code to perform the same task over and over again.

Ways to Address Memory

- Machine code comprises an operand and opcode.
- Operand is the value relating to the data on which the instruction should be performed.
- Opcode holds the instruction and the addressing mode.
- The addressing mode is how the operand should be interpreted.

Addressing Modes

- Immediate Addressing – The operand is the value itself and the instruction is performed on it.
- Direct Addressing – The operand provides the address of the value the instruction should be performed on.
- Indirect Addressing – The operand holds the address of a register. The register holds the address of the data.
- Indexed Addressing – An index register stores a certain value. The address of the operand is found by adding the index register and the operand.

	Merits	Drawbacks	Uses
Waterfall	• Straightforward to manage • Clearly documented	• Lack of flexibility • No risk analysis • Limited user involvement	Static, low-risk projects with little user input.
Agile	• High quality code • Flexible to changing requirements • Regular user input	• Poor documentation	Small to medium projects with unclear initial requirements.
Extreme Programming	• High quality code • Constant user involvement means high usability	• High cost as two people are needed • Teamwork is essential • User needs to be present	Small to medium projects with unclear initial requirements requiring excellent usability.
Spiral	• Thorough risk-analysis • Caters to changing user needs • Prototypes produced throughout	• Expensive to hire risk assessors • Lack of focus on code efficiency • High costs due to constant prototyping	Large, risk-intensive projects with a high budget.
Rapid Applicatior Development	• Caters to changing requirements • Highly usable finished product • Focus on core features, reducing development time	• Poorer quality documentation • Fast pace and late changes may reduce code quality	Small to medium, low-budget projects with short time-frames.

Assembly Language

- One level up from machine code.
- Low level language.
- Uses abbreviations for machine code called mnemonics.
- Processor specific.
- One line in assembly language equals one line in machine code.

Mnemonic	Instruction	Function
ADD	Add	Add the value at the memory address to the value in the Accumulator
SUB	Subtract	Subtract the value at the memory address from the value in the Accumulator
STA	Store	Store the value in the Accumulator at the memory address
LDA	Load	Load the value at the memory address to the Accumulator
INP	Input	Allows the user to input a value to be held in the Accumulator
OUT	Output	Prints the value in the Accumulator
HLT	Halt	Stops the program at that line
DAT	Data	Creates a flag with a label at which data is stored.
BRZ	Branch if zero	Branches to an address if the value in the Accumulator is zero. A conditional branch.
BRP	Branch if positive	Branches to a given address if the value in the Accumulator is positive. A conditional branch.
BRA	Branch always	Branches to a given address no matter the value in the Accumulator. An unconditional branch.

Development Methodologies

Extreme Programming

- An agile model.
- Development team includes developers and user representatives.
- The system requirements are based on “user stories”.
- Produces highly usable software and high quality code.
- Programmers work no longer than 40 hours per week.
- Hard to produce high quality documentation.

Rapid Application Development

- An iterative methodology.
- Uses partially functioning prototypes.
- Users trial a prototype.
- Focus groups gather user requirements.
- This informs the next prototype.
- This cycle repeats.
- Used where user requirements are unclear.
- Code may be inefficient.

Spiral Programming

- Used for high risk projects.
- Has four stages:
- Analyse requirements.
- Locate and mitigate risks.
- Develop, test and implement.
- Evaluate to inform the next iteration.
- The project may be terminated if it is deemed too risky.
- Specialist risk assessors are needed.

Agile Methodologies

- A collection of mythologies.
- Aimed to improve flexibility.
- Adapt quickly to changing user requirements.
- Sections of the program are developed in parallel.
- Different stages of development can be carried out simultaneously.
- A prototype is provided early and improved in an iterative manner.
- Low focus on documentation.
- High focus on user satisfaction.

Waterfall

- The stages are completed in order.
- The clear structure makes this model easy to follow.
- Changes mean that all stages must be revisited.
- User involvement is low.

Unit 1.2 Software and Software Development (Page 2)

Procedural Programming

- Simple to implement.
- Applicable to many problems.
- Is not suited to every problem.
- Uses traditional data types and structures.

Structured Programming

- A subsection of procedural programming
- The flow is given four structures: sequence, selection, iteration and recursion.

Object Orientated Programming

Advantages

- Reusable
- Code is more reliable
- Code is easy to maintain and update
- Classes can be reused, saving time and effort

Disadvantages

- Requires an alternative style of thinking
- Not suited to every problem
- Not best suited for small problems

Attributes, Methods, Classes and Objects.

- **Class** – a template for an object. Defines the behaviour and state of the object.
- **State** – defined by attributes giving the object’s properties.
- **Behaviour** – defined by the methods. Describes the action an object can perform.
- **Instantiation** – using a class to create an object.
- **Object** - an instance of a class. Classes can create multiple objects.
- **Setter** – a method which sets the value of an attribute.
- **Getter** – a method which retrieves the value of an attribute.
- **Constructor method** – Allows a new object to be created from a class. Every class must have one.
- **Inheritance** - process where a subclass will inherit all methods and attributes of a superclass.
- **Polymorphism** – allows objects to behave differently depending on their class.
- **Overloading** – avoiding a method by passing different parameters to a method.
- **Overriding** – redefining a method to allow it to produce a different output or function differently.

Programming Paradgims

Imperative
Code specifies clearly the actions to be performed

- **Procedural**
  - Widely used
  - Can be applied to a wide range of problems
  - Written as a set of instructions which are carried out one by one
  - Easy to write
- **Object Orientated**
  - Useful for problems which can be broken down into distinct components
  - Based around objects which have attributes and methods
  - Focus on reusability, maintainability and ease of updating

Declarative
States the desired result with the language deciding how to acchieve it. Users are unaware how the result is obtained

- **Functional**
  - Based around functions
  - Function calls can be combined
  - Linked to mathematics
- **Logic**
  - Rules and facts of the problem are defined
  - Queries are used to find answers to these problems