

Computer Architecture

- **Von Neumann Architecture** has one control unit, ALU, registers and memory unit with a shared memory and data bus used for data and instructions.
- **Harvard Architecture** has separate memories for instructions and data. It is more commonly used in embedded processors
- Von Neumann Architecture is cheaper to develop as the control unit is simpler and allows programs to be optimised in size.
- Harvard Architecture allows data and instructions can be fetched in parallel and both memories can be different sizes.

Contemporary Processing

- Combines Harvard and Von Neumann architecture
- Von Neumann is used when working with data and instructions in main memory
- Harvard is used when working with cache.
- Has a separate instruction and data cache.



RAM and ROM

Random Access Memory (RAM)

- Volatile
- Holds data and programs which are currently in use
- High access speeds
- Very expensive per gigabyte

Read Only Memory (ROM)

- Non-volatile (Cannot be modified)
- Used to store fixed instructions such as the computer start up routine

Fetch Decode Execute Cycle and Registers

- The order operations take place to execute an instruction.
- **Fetch Phase:**
 - Address copied from the PC to the MAR.
 - Data bus copies the instruction from that location to the MDR
 - At the same time, the contents of the PC increase by 1
 - The value is then copied from the MDR to the CIR
- **Decode Phase:**
 - The contents of the CIR are split into operand and opcode
- **Execute Phase:**
 - The opcode is executed on the operand.

Busses and Assembly Language

- Assembly code uses mnemonics to represent instructions.
- Instructions are divided into operand and opcode
- Opcode is the type of instruction and the hardware to execute it.
- Operand is the address where the operation is performed.

Multi-core and Parallel Systems

- Multi-core CPUs have many cores which complete separate fetch-execute cycles independently.
- Parallel systems can carry out multiple instructions simultaneously using a single core using techniques like pipelining.

Unit 1.1 The Characteristics of Contemporary Processors, Input, Output and Storage Devices

CPU Components

- The **ALU** (Arithmetic and Logic Unit) carries out arithmetical and logical operations.
- The **CU** (Control Unit) directs operations inside the processor.
- **Registers** are small, fast memory cells used to temporarily store data.

Program Counter (PC)	Stores the address of the next instruction to be executed.
Accumulator (ACC)	Stores the results of calculations.
Memory Address Register (MAR)	Holds the address in memory that is to be written to or read from.
Memory Data Register (MDR)	Holds data which has been read or needs to be written.
Current Instruction Register (CIR)	Stores the current instruction, split into operand and opcode.

- Buses are parallel wires connecting two or more CPU components together.
- The number of parallel wires determines the bus width.
- The system bus contains the data bus, control bus, and address bus.

Data Bus	A bi-directional bus which transfers data and instructions between components.
Address Bus	Transmits the location in memory where data should be read or written.
Control Bus	A bi-directional bus which transmits control signals.

Flash Storage

- Fast and compact
- Logic gates store an electrical charge
- High represents a binary 1
- Low represents a binary 0
- Information is stored in blocks which are combined to form pages
- More expensive
- Limited lifespan

Solid State Drives

- Light and portable
- No moving parts
- More resistant to damage from movement than hard disk drives
- High data transfer rates
- Smaller capacity than hard disk drives

Magnetic Storage

- Two magnetic states represent binary
 - Polarised sectors represent 1
 - Unpolarised sectors represent 0
- Can be damaged by strong magnets

Hard Disk Drives

- High capacity
- Magnetic platters rotate at high speeds beneath a read/write head
- Multiple platters are stacked to maximise storage capacity
- Moving parts can become damaged

Magnetic Tape

- An older storage medium
- Tape is round onto reels within a cartridge.
- The tape drive spins the reels to move the tape across a reader

Floppy Disks

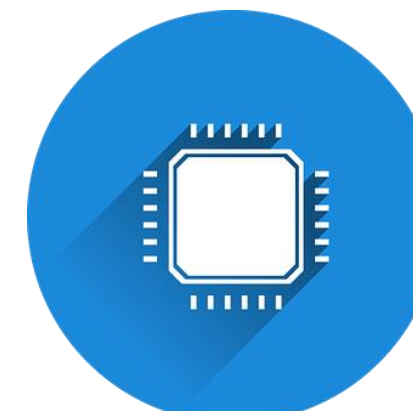
- A thin magnetic disk in a plastic case.
- Small and portable
- Typical storage capacity of 1MB

Virtual Storage

- A method of storing information remotely.
- Allows multiple computers to access data over a network or The Internet.
- Includes cloud storage and network accessible storage.
- Becoming more popular as network and Internet speeds increase.
- Relies on a network connection for access to data.
- Limited by network speed.

Input, Output and Storage Devices

- Input devices are used to send data to the computer, such as a keyboard, mouse or sensor.
- Output devices allow the computer to send information out, such as a speaker or screen.
- Storage devices allow data to be stored such as a hard drive.
- Some devices can be both an output and input device, such as a touch screen.
- Factors such as speed, accuracy, cost and relevance to the task should be considered when choosing devices.



Optical Storage

- Use lasers to read and write to a disk.
 - Sectors of the disc are written in a spiral.
 - Pits scatters light representing 0
 - Lands reflects light representing 1
 - Small and light so very portable
 - Easily scratched
 - Not very fast
- Compact Disk (CD)**
- Commonly used for audio but can store any data type
 - Stores relatively little information
- Digital Versatile Disc (DVD)**
- Higher storage capacity than CDs
 - Often used to store videos
- Blu-Ray**
- More than five times as much storage as a DVDs
 - Used to store HD films

Factors Affecting CPU Performance

Clock Speed:

- Determined by the system clock
- All activities begin on a clock pulse
- Each operation starts when the clock changes from 0 to 1
- The clock speed is the number of clock cycles which can be completed in a second.
- Faster clock speed = better performance

Number of Cores:

- Each core is an independent processor which executes its own fetch-execute cycle
- CPUs with several cores can complete more than one fetch-execute cycle at the same time
- Some applications can only use one core.
- More cores = better performance

Amount and type of Cache Memory

- Cache memory is fast memory built into the CPU
- Instructions are held in cache allowing them to be accessed quickly if needed.
- As cache fills up, unused instructions are overwritten.
- More cache = better performance
- Cache can be Level 1, 2, or 3
- Level 1 is the fastest but smallest
- Level 3 is the slowest but largest

Reduced Instruction Set Computers (RISC)

- Small instruction set
- One instruction is one line of machine code
- Used in personal computers

Complex Instruction Set Computers (CISC)

- Large instruction set
- Instructions built into hardware
- Used in microcontrollers and embedded systems
- Compiler has less work to do
- Less RAM is needed to store the code

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

Operators

Arithmetic operators carry out maths functions

- Examples include +, * and –
- ** is exponentiation, meaning power of
- DIV or // is integer division, meaning division ignoring any remainder
- MOD or % finds the remainder only of a division

Relational operators make comparisons between values

- Examples include <, > and !=

Boolean operators carry out logical functions

- Examples include AND, OR and NOT



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

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 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.

File handling

Open a file for reading

```
myFile = openRead("filename.txt")
```

Read a line from a file

```
fileContent = myFile.readLine()
```

Close a file

```
myFile.close()
```

Open a file for writing

```
myFile = openWrite("filename.txt")
```

Write a line to a file

```
myFile.writeLine("Hello World")
```

Program Flow

Sequence

- Code is executed one line at a time top down

Selection

- Blocks of code run only if a certain condition is met.
- IF, ELSE IF and ELSE statements are used to control execution.

Iteration

- Blocks of code are repeated multiple times based on a number or condition.
- FOR, WHILE, and REPEAT UNTIL statements are used to control execution.

Variables and Constants

- Named memory locations used to store data.
- The content of variables can be changed during execution.
- The content of constants cannot be changed during execution.

Device Drivers

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

Procedures and Functions

- Named blocks of code which perform a specific task.
- Functions always return a value, procedures may or may not.

	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.

String handling

Get the length of a string - stringname.length

Get a section of a string (substring) - stringname.substring(start, numberOfCharacters)

Assembly Language

- Low level language, one level up from machine code
- Uses mnemonics (abbreviations)
- Processor-specific

Mnemonic	Instruction	Function
ADD	Add	Add the value at a memory address to the value in the Accumulator
SUB	Subtract	Subtract the value at a memory address from the value in the Accumulator
STA	Store	Store the value in the Accumulator at a memory address
LDA	Load	Load the value at a memory address into the Accumulator
INP	Input	Store user input in the Accumulator
OUT	Output	Prints the value currently in the Accumulator
HLT	Halt	Stops the program and prevent further execution
DAT	Data	Creates a flag with a label at which data is stored.
BRZ	Branch if zero	Branches to a given address if the value in the Accumulator is zero.
BRP	Branch if positive	Branches to a given address if the value in the Accumulator is positive.
BRA	Branch always	Branches to a given address whatever value is in the Accumulator.

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.

Translators

- Covert source code into object code.

Compiler

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

Interpreter

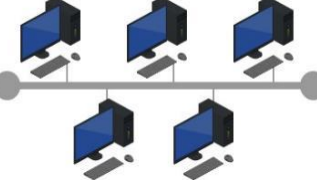
- Translates and executes code line by line.
- Will error if a line contains an error.
- Slower to run than compiled code.
- Code is platform independent.
- Useful for testing.

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.

Switching Methods Circuit Switching • A communication method using a direct link between two devices. • The link is maintained for the entire transfer. • The devices must send and receive data at the same rate. Packet Switching • The sending of data packets across a network. • Data is broken down into smaller packets before being sent. • Packets may take different routes across the network.	CSS • Cascading Style Sheets • A language used to describe the style and formatting of the page. • Internal (embedded) CSS places the style tags directly in the HTML file. • External CSS places the tags within a separate style sheet linked to the page.	DNS • Domain Name System • Allows a meaningful name to be assigned to network resources • DNS Servers translate domain names to IP addresses		Data Packets • Data is broken down into smaller parts called packets before being sent. Header • Holds the sender and recipient's IP address • The protocol being used • The order of the packet • The time to live or hop limit Payload • The actual data being sent Trailer • Holds the checksum or cyclic redundancy check	JavaScript • Adds interactivity and movement to websites. • Similar to Python. • Is interpreted not compiled, allowing it to be changed in the browser at run time. • Can be used to dynamically change the page as it is displayed or to send data to the server. • Allows the local computer to fix invalid data before sending. • Reduces the load on the server
	Client Server Network • Clients connect to a central server. • The server is a powerful computer central to the network. • It holds all the data. • More secure setup. • Clients do not need to be backed up. • Data and resources can be shared easily. • Expensive to setup. • More secure.	Peer to Peer Network • Computers are connected directly to each other. • Computers share data with one another. • Quick, cheap and easy to setup. • Less secure. • Easier to maintain.			

Unit 3: Exchanging Data

Databases • An entity is item about which information is stored such as books, or customers. • Attributes are the categories in which data is collected such as height or name. Flat File Database • Consists of a single file. • Usually based around a single entity. • Only one table. Relational Database • Uses many tables to store data about different entities. • These tables are linked together. Primary Key • A unique identifier, different for each object in the database. Foreign Key • Used to link two tables together. • The primary key from a different table. Secondary Key • Used to enable searching or sorting. • Usually a common field like name.	Bus Network  • All devices are connected to a single cable (called the bus) • A terminator is at each end of the cable. Advantages: • Easy to add extra devices. • Cheap to install as it doesn't require much cable. Disadvantages • If the cable fails or is damaged the whole network will fail. • Performance becomes slower as additional devices are connected due to data collisions. • Each device receives all data, a security risk	Star Network  • All nodes are connected to one or more central switches. • Often used with wireless networks, where a Wireless Access Point or WAP will be the central connection Advantages: • Every device has its own connection so failure of one node will not affect others. • New devices can be added by connecting them to the switch. • Usually have higher performance as a message is passed only to its intended recipient. Disadvantages: • If the switch fails it takes out the whole network. • Requires a lot of cable so can be expensive.	Selecting, Managing and Exchanging Data • Data may be selected based around set criteria • Only data matching the criteria is input to the data • SQL can be used to sort, structure and filter the data • Data may need to be transferred between systems or organisations • This is know as data exchange • This can be accomplished using EDI (Electronic Data Exchange)	Computer Networks • A network is two or more computers connected together for the purposes of transmitting data. • The physical topology defines the physical layout of the network • The logical topology defines the way data flows through the network • A protocol is a set of rules for communication between devices. • They allow devices from different vendors to communicate • A LAN (local area network) covers a small physical area. • A WAN (wide area network) covers a large physical area.	Protocol Layers • Sending data works from the top down • Receiving data works from the bottom up Application Layer • At the top of the stack • Specifies which protocol must be used in relation to the application in use. Transport Layer • Forms an end to end connection between recipient and source using TCP. • Splits the data into packers labelled with a packet number. • Requests any missing packets are retransmitted. Network Layer • Adds source and destination IP Addresses to the packets • Routers operate at this layer using the IP Address to forward packets. Link Layer • The lowest layer. • The physical connection between devices. • Adds the MAC Address to identify the NIC of the source and destination computers.
Capturing Data • There are many ways to capture the data needed for a database. • The most appropriate way will depend on the type and quantity of data needed and available resources. • Data may be manually entered by a human or scanned in using optical character recognition, sensors or barcodes.	HTML • HyperText Markup Language • The language web pages are written in. • Describes the structure and order of the web page. • Browsers interpret the code to render the page. • The body contains the main part of the web page. • The header contains additional information such as the title. • Uses tags within <> • <> opens a tag and </> closes the tag	Compression Methods. • Lossy compression reduces the size of the file whilst removing some information. • This means the original file cannot be recreated. • Lossless compression does not loose any of the original information. • This allows the original file to be recreated.	Entity Relationship Modelling • One to One – Each entity can be associated with one other entity only. • One to Many – A single table many entities in another table. • Many to Many – Many entities in one table are linked to many in another table.	Classes and Identifiers • Allows items to be grouped and styled together. • Uses <div> tags • Classes begin with a .and can be used many times • Identifiers begin with a # and must be unique	HTML Tags • <html> - Indicates the code is html • <body> - Marks the body of the page • <link> - Links to a CSS Stylesheet • <head> - Marks the page header • <title> - Marks the page title • <h1> , <h2>, <h3> - Marks headings • <p> - Marks a paragraph • - An Image • link text - a link • - An ordered list • - An unordered list • - An item in a list • <div> - Divides the page into separate areas

High Level Programming Languages

- Much easier to learn, write and debug.
- Examples include Python, Java and C
- Code written in these languages must be translated to machine code before it can be executed.

Advantages

- Much more widely understood and used.
- Easier to learn, code in and understand.
- Much quicker to produce usable code.
- More support and learning resources are available.
- Easier to debug and find issues

Disadvantages

- Less flexible.
- Must be translated before being executed
- Very difficult to write and understand.
- Much more time consuming to produce code.

Stacks and Queues

- Stacks
 - Last in first out
 - Items can only be added or removed from the top
 - Used for back or undo buttons
 - Can be dynamic or static structure
- Queues
 - First in first out data structure
 - Items are added at the beginning and removed at the end
 - Used in printers and keyboards
 - Linear queue with items added into the next space
 - Space inefficient
 - Uses pointers at the front and back
 - Circular queues have a rear pointer that can loop back to the beginning to use empty space.

```
content: "";
content: none;
}
table {
  border-collapse: collapse;
  border-spacing: 0;
}
button, input, select, textarea { margin: 0 }
:focus { outline: 0 }
alink { -webkit-tap-highlight-color: #FF5E99 }
img, video, object, embed {
  max-width: 100%;
  height: auto!important;
}
iframe { max-width: 100% }
blockquote {
  font-style: italic;
  font-weight: normal;
  font-family: Georgia,Serif;
  font-size: 15px;
  padding: 0 10px 20px 27px;
  position: relative;
  margin-top: 25px;
}
blockquote:after {
  position: absolute;
  content: "";
}
blockquote p { margin-bottom: 10px }
strong, b { font-weight: bold }
em, i, cite {
  font-style: normal;
  font-family: arial;
}
small { font-size: 100% }
figure { margin: 10px 0 }
code, pre {
  font-family: monospace,consolas,sans-serif;
  font-weight: normal;
  font-style: normal;
}
pre {
  margin: 5px 0 20px 0;
  line-height: 1.3em;
  padding: 8px 10px;
  overflow: auto;
}
pre {
  margin: 5px 0 20px 0;
  line-height: 1.3em;
  padding: 8px 10px;
  overflow: auto;
}
```

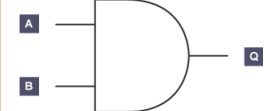


Karnaugh Maps

- Used to simplify Boolean expressions
- Can be used for truth tables with between two and four variables
- Values in columns and rows must be written using grey code
- Columns and rows only differ by one bit
 - 1) Write the truth table as a Karnaugh Map
 - 2) Highlight all the 1s
 - 3) Only groups of 1s with edged equal to a power of 2 may be highlighted
 - 4) Remove variables which change within the highlighting
 - 5) Keep variables which do not change

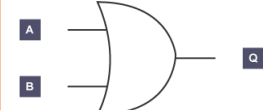
Boolean Operators

AND - two conditions must be met for the statement to be true
Written as AND or .



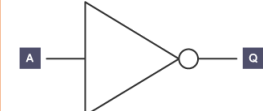
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

OR - at least one condition must be met for the statement to be true
Written as OR or +



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

NOT - inverts the result, e.g. NOT(A AND B) will only be false when both A and B are true
Written as NOT or -



A	Q
1	0
0	1

XOR - Also know as Exclusive OR. Works the same as an OR gate, but will output 1 only if one or the other and not both inputs are 1.
Written as XOR or ⊕



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Normalisation

- Maximises the precision in any number of bits.
- Adjust the mantissa so that it begins with 01 for positive numbers and 10 for negative numbers.

Unit 1.4 Data Types, Data Structures and Algorithms

Data Structures

- Records
- A row in a file or table
- Widely used in databases
- Made up of fields
- Lists
- A number of items
- Items can occur more than once
- Data can be of more than one data type
- Tuples
- An ordered set of values
- Cannot be changed once initialised
- Initialised with regular rather than square brackets
- Arrays
- An ordered set of elements, each of the same type.
- A 1D array is like a list.
- A 2D array is like a table.
- A 3D array is like a multi page spreadsheet.
- 2D arrays are searched first by the rows and then the columns.

Floating Point Numbers

- Similar to scientific notation
- Numbers are split into Mantissa and Exponent
- The mantissa has the binary point after the most significant bit
- Then convert the exponent to decimal
- Move the binary point according to the exponent

Character Sets

- A collection of codes and their corresponding characters.
- ASCII**
 - American standard code for information interchange
 - Older character set
 - Uses 7 bits representing 27 (128) characters
 - Insufficient characters to represent multiple languages
- Unicode**
 - Developed in response to ASCII's limited characters
 - Varying number of bits allows over 1 million characters
 - Many characters yet to be used
 - Includes different symbols and emojis

Negative Numbers in Binary

Sign Magnitude

- A leading 1 is added for a negative number.
- A leading 0 is added for a positive number.
- Two's Complement**
 - Makes the most significant bit negative.
 - Converting to Two's Complement involves flipping all the bits in the positive version of the number and adding one.
 - Makes arithmetic with negative numbers easier.

Positive Integers in Binary

- Each binary digit is called a bit
- Eight bits form a byte
- Four bits (half a byte) is called a nybble
- The most significant bit is furthest left
- The least significant bit is furthest right

Data Types

Integer

- A whole number
- May be positive, negative or 0
- Cannot have a fraction or decimal point
- Often used for counting objects
- e.g. 5, -1, 0, 10

Real

- Positive or negative number
- May have a decimal point
- Often used for measurements
- e.g. 5, -10, 100.556, 15.2

Character

- A single symbol
- May be a letter, number or character
- Uppercase and lowercase letters are different characters
- e.g. A, a, 5, M, ^, @

String

- A collection of characters
- Can store one or many strings
- Often used to contain text
- Leading 0s are not trimmed so useful for storing phone numbers

Boolean

- True or False only

Binary Addition

- 0 + 0 = 1
- 0 + 1 = 1
- 1 + 1 = 10
- 1 + 1 + 1 = 11

Stack and Queue Operations

Stacks

- isEmpty() - Checks if the stack is empty
- push(value) - Adds a new value to the top of the stack
- peek() - Returns the top value of the stack
- pop() - Returns and removes the top value of the stack
- size() - Returns the size of the stack
- isFull() - Checks if the stack is full
- Queues**
 - enqueue(value) - Adds a new item at the end of the queue
 - dequeue() - Removes the item at the end of the queue
 - isEmpty() - Checks if the queue is empty
 - isFull() - Checks if the queue is full

Hexadecimal

- Base 16.
- Characters 0-9 are used as usual.
- A-F are used instead of 10-15.
- Place values begin with 1 and increase in powers of 16

Converting Hexadecimal to Binary

- Convert each digit to a decimal number
- Convert these to a binary nybble
- Join the nybbles into a single binary number

Converting Hexadecimal to Decimal

- Convert to binary
- Convert the binary to decimal

Trace Tables

- A method of recording the values used within an algorithm at each stage of processing to help in troubleshooting
- Tests algorithms for logic errors which occur when the algorithm is executed.
- Simulates the steps of algorithm.
- Each stage is executed individually allowing inputs, outputs, variables, and processes to be checked for the correct value at each stage.
- A great way to spot errors

	Stage	X	Y	Output
	1	3	1	
	2		2	
	3	2		
X = 3 Y = 1 while X > 0 Y = Y + 1 X = X - 1 print(Y)	4		3	
	5	1		
	6		4	
	7	0		
	8			4

Combining and Manipulating Boolean Operations

- Boolean operators can be combined to form Boolean equations
- This follows the same way as combining standard maths operators
- The equation can be represented by a truth table
- Sometimes a long expression can share a truth table with a shorter expression
- It is better to use the shorter version.

Binary Subtraction

- Use Two's Complement.
- Use the same rules as adding a negative number.
- Use binary addition with a negative two's complement number.

The Computer Misuse Act 1990

- Covers hacking and other malicious use of computers
- Frequently updated as technology changes
- Contains three primary offenses:
 1. Unauthorised access to computer materials.
 2. Unauthorised access with intent to commit further offences.
 3. Unauthorised modification of computer material.

Computers in the Workforce

- Computers increase efficiency and productivity.
- This reduces labour costs giving lower prices to consumers.
- Computers can easily conduct repetitive and tedious roles, reducing the need for humans to.
- This can contribute to rising unemployment.
- Manufacturing roles have been badly hit by this
- There is a rise in services being offered online only.
- This reduces costs to the business.
- This can exclude those without online access.



Analysing Personal Information

- Large amounts of data from different sources can be pulled together. This is known as big data.
- This can identify patterns or other connections.
- This data can reveal a lot about individuals and their behaviour.
- It can also be used to target advertisements.
- Many people argue large companies should be more transparent about how they are doing this.
- People argue organisations have a responsibility to share trends they find.

Environmental Effects

- Technology evolves quickly and is more widely accessible than ever before.
- Modern devices are not always designed to be repaired if they break.
- This results in a huge number of devices being thrown away.
- Computer components are toxic and can cause environmental damage if not correctly disposed of.
- Electronic waste is sometimes shipped to countries with lower disposal standards.
- Computers and associated servers and equipment require huge amounts of electricity to power and cool them.
- This is often generated by burning greenhouse gasses.

Layout, Colour Paradigms and Character Sets

- Web designers should consider who will be using their sites when creating them.
- The Equality Act 2010 makes it illegal to discriminate against a group of people.
- Those with visual impairments may need to enlarge text or alter contrast.
- Websites should provide alt text for images.
- Transcripts of audio should be provided for those with hearing impairments.
- Web designers should consider how colours are interpreted in different countries.
- Some colours are regarded as unlucky in certain cultures.
- Web content may need to be translated into different languages.
- Unicode is the preferred character set due to its large number of characters.

Unit 1.5 Legal, Moral, Cultural and Ethical Issues

The Regulation of Investigatory Powers Act 2000

- Covers the use of investigation and surveillance by public organisations
- Requires Internet Service Providers and mobile phone providers to provide certain information about their users to an authorised authority if requested
- Requires ISPs to install suitable monitoring hardware on their network
- Controversial as smaller organisations such as local councils can make use of its powers
- Some feel that it is improperly used and an invasion of privacy



Monitoring Behaviour

- People's computer usage may be monitored and tracked.
- Computers may be used to track people in other ways such as via CCTV.
- Employers may monitor the computer use of their employees to ensure productivity.
- These technologies are used to track and prevent crime.
- Some people believe this is unethical.
- Others argue this is essential to maintain safety and security.
- What should and should not be monitored?

Artificial Intelligence

- The ability of a computer system to behave in a way which replicates human intelligence, analysis and decision making.
- Backed by a large knowledge base.
- Used in systems designed to replicate system experts.
- Used to analyse huge datasets.
- Used in medicine to form diagnosis
- Used in neural networks for pattern and fraud identification.
- Used in voice recognition.
- Raises privacy concerns.
- What rights might a sentient AI have?
- Who is responsible when things go wrong?

Censorship

- Restricting what content people can view, publish or access.
- ISPs block illegal content such as that associated with extremism.
- Some people worry that this can be misused or used to push certain ideas or beliefs.
- Some promote a Free Internet with no filtering at all.
- Censorship may also exist at a smaller level within a workplace.
- It may be used to maintain employee productivity or maintain the security of the network.

Offensive Communications

- Unauthorised copying of software, music or other content is theft.
- The Internet has made this easy and widespread.
- The Internet increases anonymity.
- People are sharing more information than ever online.
- There has been an increase in cyber bullying and stalking.
- The Malicious Communications Act 1998 makes it an offense to send indecent or offensive messages online.

The Copyright, Design and Patents Act 1988

- Protects works such as logos, photos, software code and music.
- Applied automatically to original works.
- Expires between 25 and 70 years after the author's death.
- Extended in 1992 to include computer software

Automated Decision Making

- Used to control what is shown first on social networking feeds.
- May create an "echo chamber" where people are not exposed to views which challenge their own.
- Used in application processes such as for mortgages or credit cards.
- Relying entirely on automated decisions could mean people are unfairly treated.
- Used in driverless cars.
- Raises many ethical concerns.
- Algorithms must be written and tested to avoid bias.

The Data Protection Act 1998

- Applies to data stored electronically and on paper
- Covers personal data, defined as data which either alone or in conjunction with other data can be used to identify a living person.
- Any individual who can be identified by personal data is known as the data subject.
- Superseded in 2018 by the European General Data Protection Regulations

Eight specific protections are provided for data subjects

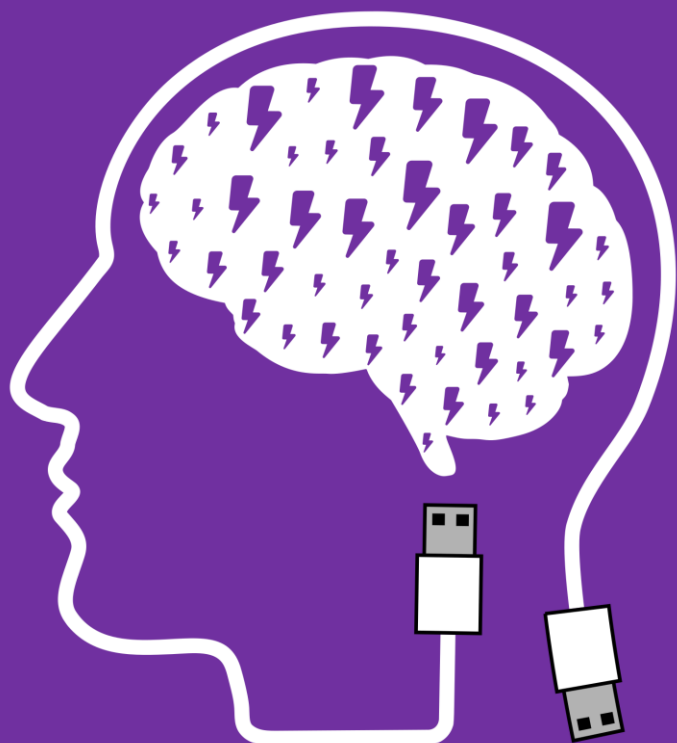
1. Personal data shall be processed fairly and lawfully.
2. Personal data shall be obtained only for one or more specified and lawful purpose.
3. Personal data shall be adequate, relevant and not excessive for its purpose.
4. Personal data shall be accurate and up to date.
5. Personal data shall not be kept for longer than is necessary.
6. Personal data shall be processed in accordance with the rights of data subject.
7. Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.
8. Personal data shall not be transferred to a country or territory outside the European Economic Area, unless that country or territory ensures an adequate level of data protection

The act provides three rights to data subjects:

1. The right to request a copy of the data held about them
2. The right to correct inaccurate data held about them
3. The right to prevent the use of their data for marketing purposes

Preconditions

- Things which are needed before the program can run.
- The code expects the information passed to it to meet certain criteria.
- The code may test for these when it is run.
- They may instead be included within documentation.
- Including this information within documentation reduces the complexity of the program and makes it easier to use.
- Preconditions make it easier to reuse subroutines.



Abstraction

- Removing unnecessary detail.
- Representing the key elements of the problem.
- Must consider what information is actually relevant to the problem at hand.
- Complex problems can be split into several layers of abstraction.
- Higher layers are closer to the user, possibly providing a user interface.
- Lower levels interact with the computer.

Abstraction and Reality

- Abstraction is more simplified than reality.
- Real world items are simplified into computer structures such as a table, variable or database.
- Objects used in object oriented programming can be an abstraction of real world entities.
- Attributes can represent the characteristics of a real world object.
- Methods can represent the actions a real world object may perform.

Creating an Abstraction Model

- What problem needs to be solved?
- Who will use the model?
- How will the model be used?
- Which are the key elements of the problem for the people using the model and how they will use it?

Abstraction by Generalisation

- Similar elements of a problem may be grouped together.
- This allows common problems to be categorised.
- They can then be solved with a common solution.

Procedural Abstraction

- Allows a programmer to use a function without understanding the detail of its implementation.
- Used with data structures and in decomposition.
- Models the purpose of a subroutine without considering how it does what it does.

Unit 2.1 Elements of Computational Thinking

Inputs and Outputs

- An input is any data required to solve the problem.
- These may be entered by the user, or obtained from hardware such as a sensor.
- Outputs are the solutions to the problem which are returned.
- They can only be produced once the input has been processed.
- It is important to consider the methods used to capture data from the user and to present it back to them.
- Think about the data structures used.
- Think about the devices used.
- Think about what outputs are needed first.
- Use this information to consider what inputs are needed to produce the required output.

Conditions Affecting a Decision

- Effectiveness
- Convenience
- Cost
- Efficiency
- Relevance
- Available skills and resources
- All these conditions are important.
- Some may be more important to a particular decision.



The Order of Steps

- It is important to consider the order in which operations are performed.
- Certain inputs may be required before processing.
- Inputs may need to be validated, this must occur after the input is received and before it is processed.
- It may be possible for several subroutines to be executed at the same time.
- Also consider how subroutines interact with one another.
- Code should be written to prevent operations occurring in an order which would cause an error or prevent the program from functioning as intended.

Problem Decomposition

- Breaking down a large problem into smaller parts.
- These smaller parts are easier to solve.
- The smaller parts are easy to divide among a team.
- Top down design, also called stepwise refinement is often used to do this.
- This technique divides a problem into levels of complexity.
- Problems are broken down over and over until each problem is a single task.
- Each task can then be solved with a single subroutine.
- Subroutines can be tested and developed separately.
- Consider how each subroutine is implemented.
- The subroutines need to be joined to form the whole solution.
- Start with the lowest level components and work up.
- Some tasks may be solved with an existing module or library.

Reusable Program Components

- Common functions can be packaged into a library.
- This makes it easier to reuse them throughout a project.
- Abstract data structures, subroutines and classes can all be reused in this way.
- Decomposition is used to indicate where components of an existing program can be reused.
- Reusable components have already been tested and so are more reliable.
- They make development less time consuming and therefore less costly.

The Need for Abstraction

- Allows those who are not experts in a field to use systems by hiding more complex information which is irrelevant to using the system.
- Allows more efficient design by encouraging focus on the core elements of a problem.
- Reduces the time spent on a project.
- Prevents a project becoming too large or complex.
- Low-level programming languages directly interact with hardware but are hard to write so high-level languages abstract the machine code that is executed when a program is run.
- The TCP/IP model is an example of abstraction in networking.

Data Abstraction

- Programmers may use complex data structures without needing to understand how they are implemented in detail.
- How is the data being stored and filtered?

Decision Making

- There are many decisions involved with making and designing programs.
- It is important to consider these decisions carefully.
- Often, the available choices for a decision may be limited, simplifying the decision.
- Identifying the decisions which need to be made allows information to be gathered on potential choices.
- In flow charts, decisions are represented by diamonds.

Decisions Affecting Program Flow

- There may be many routes through a program.
- Decisions by the user will affect the route taken.
- It is important to identify places where the user will need to make a decision and plan for the decisions they may make.

Testing

- Pinpoints any flaws or bugs in software.
- Ensure the software produces the correct output for an appropriate input.
- Normal data is data which the program would normally handle within the range and data type expected.
- Boundary data falls at the very edge of the valid range.
- Erroneous data falls outside the valid range and should be rejected.

Variables

- Variables can be either global or local scope.
- Scope refers to the section of code where the variable can be accessed.
- A local variable in a subroutine has precedence over a global variable with the same name.

Local Variables

- Can only be accessed within the subroutine where they were defined.
- Multiple variables with the same name can exist in different subroutines.
- Are deleted when the subroutine ends.
- Ensures subroutines are self contained.

Global Variables

- Can be accessed through the whole program.
- Used for values needed throughout the program.
- A risk the variable can be unintentionally edited.
- Uses memory for longer.

Integrated Development Environment

- Programs used to write code.
- Contains a set of tools which make it easier for programmers to write, develop and debug code.
- May include stepping, variable watching, breakpoints, source code editor and debugging tools.



Software Design Lifecycle Stages

- Analysis
- Design
- Development
- Testing
- Implementation
- Evaluation
- Maintenance

Modularity

- Large or complex programs can be split into smaller self contained modules.
- This makes it easier to divide tasks between a team and manage the project.
- It simplifies maintenance since each component can be handled individually.
- It improves the reusability of code.
- Top Down (Stepwise) Refinement
- A technique used to modularise programs.
- The problem is broken into sub problems until each sub problem is a single task.
- Modules form blocks of code called subroutines.

Programming Constructs

- Sequence – Code is executed line by line from the top down.
- Breaching – A block of code is run only if a condition is met using IF and ELSE statements
- Count Controlled Iteration – A block of code is run a certain number of times. Uses FOR, WHILE or REPEAT UNTIL statements.
- Condition Controlled Iteration – A block of code is run while or until a condition is met. Uses FOR, WHILE or REPEAT UNTIL statements.

Unit 2.2 Problem Solving and Programming

Waterfall

Merits	Drawbacks	Uses
• 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 Application 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.
--	---	--

404
Not Found

Test Strategies

Alpha Testing

- Carried out by the software development team.
- Early stage of testing.
- Bugs are pinpointed and resolved.

Beta Testing

- Carried out by end users.
- Feedback used to resolve bugs and inform development.

White Box Testing.

- Carried out by the software development team.
- Considers the program code and internal structure.
- All routes through the program are tested.

Black Box Testing

- Testers do not have access to the code or internal program structure.
- Test plans are used to trace inputs and outputs.

Dry Run Testing

- The software development team work through the code producing a trace table.
- This records the variables used and when they are updated.

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.

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.

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.

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.

Functions and Procedures

- Named code blocks which perform a particular task.
- Functions must always return a single value.
- Procedures do not have to return a value.
- Parameters can be passed to them by either reference or value.

Passing by Reference

- The address of the parameter only is given to the subroutine.
- The subroutine works on the value at the given address.

Passing by Value

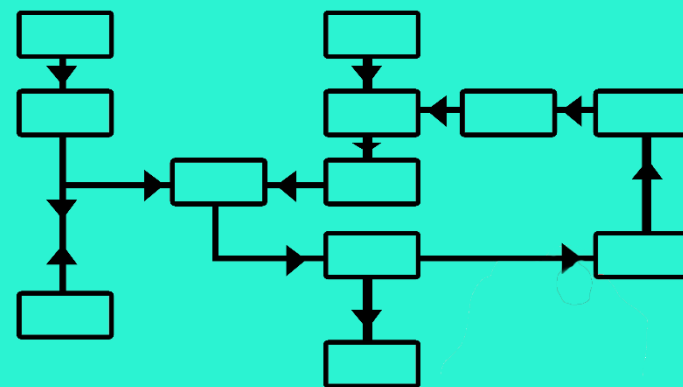
- A copy of the value is passed to the subroutine.
- The original value is unchanged.
- The copy is deleted at the end of the subroutine.
- Exam questions will use this technique unless told otherwise.
- Exam questions will use the format function(x:value, y:value)

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.

Designing Algorithms

- The priority for an algorithm is to achieve the given task.
- The second priority is to reduce time and space complexity.
- There may be a conflict between space and time complexity and the requirements and situation for an algorithm will dictate which is more important.
- To reduce space complexity, make as many changes on the original data as possible. Do not create copies.
- To reduce time complexity, reduce the number of loops.



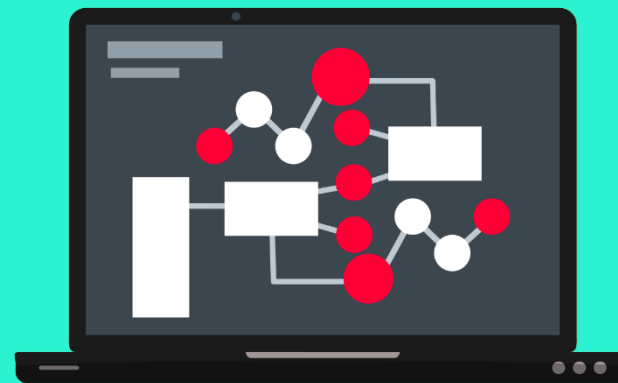
Queues

- FIFO (First in first out)
- Often an array.
- The front pointer marks the position of the first element.
- The back pointer marks the position of the next available space.
- **Queue Functions**
- Check size `size()`
- Check if empty `isEmpty()`
- Return top element (but don't remove) `peek()`
- Add to the queue `enqueue(element)`
- Remove element at the front of the queue and return it `dequeue()`

Big-O Notation

- $O(1)$ - Consistent time complexity - The amount of time is not affected by the number of inputs.
- $O(n)$ - Linear time complexity - The amount of time is directly proportional to the number of inputs.
- $O(n^2)$ - Polynomial time complexity - The amount of time is directly proportional to the number of inputs to the power of n .
- $O(2^n)$ - Exponential time complexity - The amount of time will double with every additional input.
- $O(\log n)$ - Logarithmic time complexity - The amount of time will increase at a smaller rate as the number of inputs increases.

Unit 2.3 Algorithms



Sorting Algorithms

- Places elements into a logical order.
- Usually numerical or alphabetical.
- Usually in ascending order.
- Can be set to work in descending order.

Bubble Sort

- Compares elements and swaps as needed.
- Compares element 1 to element 2.
- If they are in the wrong order, they are swapped.
- This process is repeated with 2 and 3, 3 and 4, and so on until the end of the list is reached.
- This process must be repeated as many times as there are elements in the array.
- Each repeat is referred to as a "pass".
- Can be modified to improve efficiency by using a flag to indicate if a swap has occurred during the pass.
- If no swaps are made during a pass the list must be in the correct order and so the algorithm stops.
- A slow algorithm.
- Time complexity of $O(n^2)$

Insertion Sort

- Places elements into a sorted list.
- Starts at element 2 and compares it with the element directly to its left.
- When compared, elements are inserted into the correct position in the list.
- This repeats until the last element is inserted into the correct position.
- In the 1st iteration 1 element is sorted, in the 2nd iteration 2 are sorted etc.
- Time complexity of $O(n^2)$

Stacks

- FILO (First In Last Out)
- Often an array.
- Uses a single pointer (the top pointer) to track the top of the stack.
- The top pointer is initialised at -1, with the first element being 0, the second 1 and so on.
- **Stack Functions**
- Check size `size()`
- Check if empty `isEmpty()`
- Return top element (but don't remove) `peek()`
- Add to the stack `push(element)`
- Remove top element from the stack and return it `pop()`

Logarithms

- The inverse of an exponential.
- An operation which determines how many times a certain number is multiplied by itself to reach another number.
- $x^y = \log(x)$
- $1(20) = 0$
- $8(23) = 3$

Searching Algorithms

- Used to locate an element within a data structure.
- Many different forms exist.
- Each is suited to different purposes and data structures.

Linear Search

- Most basic search algorithm.
- Works through the elements one at a time until the requested element is found.
- Does not need data to be sorted.
- Easy to implement.
- Not very efficient.
- Time Complexity is $O(n)$

Binary Search

- Only works with sorted data.
- Finds the middle element, then decides on which side of the data the requested element is.
- The unneeded half is discarded and the process repeats until either the requested element is found or it is determined that the requested element does not exist.
- A very efficient algorithm.
- Time Complexity is $O(\log n)$

Time Complexity

- How much time an algorithm needs to solve a problem.
- Measured using big-o notation.
- Shows the amount of time taken relative to the number of inputs.
- Allows the required time to be predicted.

Space Complexity

- The amount of storage space the algorithm takes up.
- Does not have a defined notation.
- Copying data increases the storage used.
- Storage space is expensive so this should be avoided.