OCR A: A-Level Computer Science

Revision for OCR A A-Level Computer Science Spec

Section 1 - Components of a Computer

Processor and Computer Hardware

The Processor

The processor refers to any electronic component capable of performing mathematical and logical operations. While the CPU (central processing unit) is the primary general purpose processor responsible for managing the system, other specialised processors utilise the same fundamental components, but are implemented and produced differently to optimise for specific workloads.

The processor is responsible for processing and executing program instructions as per the fetch-decode-execute (FDE) cycle.

1. Fetch: The address of the next instruction is copied from the program counter to the memory address register. The fetched instruction is copied to the memory data register, and the program counter is incremented by 1. The contents of the memory data register are copied to the current instruction register.
2. Decode: The instruction held in the current instruction register is decoded by the control unit, then split into opcode and operand. The opcode determines the type of instruction, and the operand holds either the address of the data to be used (copied to the memory address register) or the actual data to be used. Data is passed to the memory data register.
3. Execute: The instruction is carried out.

Control Unit

The control unit is responsible for directing the flow of data between different components of the processor. It is responsible for all three parts of the fetch-decode-execute cycle.

Buses

A bus is a set of parallel wires connecting two or more components of a computer. There are three main buses, collectively known as the system bus:

- Address bus: The bus along which an address of RAM is sent.
- Data bus: The bus along which data is sent.
- Control bus: The bus along which a control signal (e.g. read and write) is sent.

Each bus is a shared transmission medium, so only one device can transmit along a bus at any one time.

Arithmetic Logic Unit

The arithmetic logic unit (ALU) is responsible for performing arithmetic, such as adding and subtracting, as well as multiplication and division by executing bit shifts, and logical operations on data with operators like AND, OR, NOT, and XOR. Operations are performed on, and results are stored in, the accumulator.

Registers

Registers are super-fast, low-capacity primary memory devices located inside the processor. Each register has a dedicated purpose.

- Program counter (PC): Holds the address of the next instruction to be executed. Can be modified by a branch instruction, and is otherwise incremented on every FDE cycle.
- Current instruction register (CIR): Holds the current instruction executed.
- Memory address register (MAR): Holds the address of the memory location from/to which data has to be read from/written.
- Memory data register (MDR): Temporarily holds data read from a memory address or that needs to be written to one. (A.k.a. the memory buffer register.

Cache Memory

Cache is a small amount of very fast memory located inside the processor. It stores frequently accessed instructions and data, so the processor does not have to fetch it from the RAM so often, which is located much further from the processor than cache. Reduces data transmission times thus result in greater processor performance. However, instructions and data in the cache could become outdated, harming data integrity.

Furthermore, cache tends to be low in capacity - if it's too large, more time is spent searching the cache for the desired data, reducing processor performance. For this reason, processors tend to have levels of cache. E.g. in a CPU, there is level 1 cache (low capacity, ~2-64KB) and level 2 cache (higher, ~0.256-2MB).

Clock speed

The clock speed dictates the number of FDE cycles that occur per second, affecting the speed of instruction execution. This number is so vast, it is measured in MHz (million hertz) or GHz (billion hertz). The larger this number, the more cycles, but the more power drawn to do so, resulting in more heat being generated.

Pipelining

Pipelining involves the fetching and decoding of instructions while another is executing. This uses components of the CPU that would otherwise be idle during the execution of a single instruction. Thus, three FDE cycles can be running at the same time. Take instructions A to F for example in an instruction pipeline:

Fetch	Decode	Execute
A		
B	A	
C	B	A
D	C	B
E	D	C
F	E	D

Cores

A core is an independent processing unit within a processor that contains all of the above components and techniques. Multiple cores are connected in parallel in a circuit and can communicate with each other. Since they are otherwise independent of each other, they can execute instructions simultaneously. However, the more cores there are, the more power is drawn, causing heat to be generated.

Computer Architecture

Von-Neumann

The Von-Neumann architecture involves a shared system bus for data and instructions, shared memory storage for data and instructions, an ALU, CU, and registers. Mainly used in general purpose computers.

Harvard Architecture

The Harvard architecture involves separate system buses leading to two separate memory storages for data and instructions, an ALU, CU, and registers. Mainly used in embedded systems.

Types of Processor

CISC Processor

Complex instruction set computer (CISC) processors have large numbers of instructions that perform complicated tasks. Compilers for CISC processors are thus simple, as machine code becomes short, simultaneously also requiring less RAM to run. However, the complexity of CISC processor circuits makes their production far more expensive. Furthermore, only about 20% of CISC processor instructions are actually used in compilation, so 80% of them are redundant, unnecessarily increasing cost. Furthermore, each CISC processor instruction doesn't have a fixed execution time, as they can be the equivalent of RISC processor instructions. This makes CISC processor execution times less reliable.

RISC Processor

Reduced instruction set computer (RISC) processors have far fewer instructions that are much simpler. An advantage is that the processors' circuits are far simpler, so they are cheaper to manufacture. Furthermore, each instruction takes the same amount of time to execute. However, compilers are far more complex due to more instructions being required for the same task, which also results in higher memory usage.

Co-Processor

A co-processor is an extra processor that supplements the functions of the primary processor. It's not a general purpose processor, but can perform tasks like floating-point arithmetic and graphics processing.

Graphics Processing Unit (GPU)

The GPU is a co-processor specialised for manipulating computer graphics, crypto mining, password cracking, etc. It has a massive parallel architecture with thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously. It is capable of performing SIMD (single instruction, multiple data), where a single instruction can be carried out on vast numbers of data.

Section 4 - Exchanging Data

Relational Databases and Normalisation

Core Info and Definitions

A relational database contains tables, making use of relations between them to store data with minimal data redundancy.

Normalisation is a process used to come up with the best possible database design by following a set of formal rules. It ensures:

- No data is unnecessarily duplicated.
- Data is consistent throughout the database

Transaction Processing and ACID

ACID

Atomicity	A rule that requires a transaction to be fully completed or not started at all. E.g. Rolling back changes if a system crashes mid-payment.
Consistency	Any changes to the database must retain its overall state/integrity. E.g. if an attempt is made to add students beyond maximum capacity, it fails.
Isolation	Each transaction is executed and committed to the database in isolation. E.g. record locking.
Durability	Once a transaction has been committed, it must be processed until completion. If it requires multiple actions, all of them must be completed before committing. I.e. saving changes.

Transaction Processing

Measures need to be taken in order to ensure that multiple users using a database does not harm referential integrity.

Record locking prevents simultaneous access to a record to prevent updates from being lost or inconsistencies in data arising.
However, deadlock can occur. This is when a committed process depends on data from a locked record, which is locked because of a running process that depends on another record locked by the former process. This can be solved with one of the following:

Serialisation	Ensures transactions do not overlap in time, so they cannot interfere with each other or lead to lost updates. A transaction cannot start until the previous one is complete. This is implemented with timestamp ordering.
Timestamp ordering	When a transaction starts, it is assigned a timestamp. If two processes are simultaneously affecting the same record/table, the process with the earlier timestamp is executed first. Each object in a database has a read timestamp and a write timestamp to mark when it was last read from or written to. If on write the object's read timestamp does not match the process', it knows that another process is operating on the object.
Commitment ordering	Ensures that transactions are not lost when two or more users are trying to access the same object. Transactions are ordered in terms of their dependencies on each other as well as the time they were initiated. Helps prevent deadlock.
Redundancy	Having multiple identical systems all manipulated at the same time. If one system goes down, another can instantly start being used to replace it until it comes back online.

First Normal Form

1. A table contains unique records.
2. Data in fields is atomic - i.e. no grouped data in fields, e.g. grades for three separate subject should be in three separate records, not one.

Second Normal Form

1. The table is in first normal form.
2. There are no partial dependencies - i.e. a field depends on only part of the primary key (seen in composite keys).

Third Normal Form

1. The table is in second normal form.
2. There are no transitive dependencies - i.e. fields **ONLY** depend on the primary key, not **any** other field.

Database concepts

Modelling

Core info and definitions

When creating a database, you must identify the entities. An entity is a "category of object, person, event or thing of interest to an organisation about which data to be recorded". These entities are presented as tables. Column headings are fields/attributes, and each row is a record.

Entities are described in written form as:
EntityName(Attr1, Attr2, Attr3, ...)

Referential integrity ensures that a foreign key in one table always refers to the primary key of another table. This prevents orphan records, maintains data consistency across tables, and enforces logical relationships between entities.

Keys

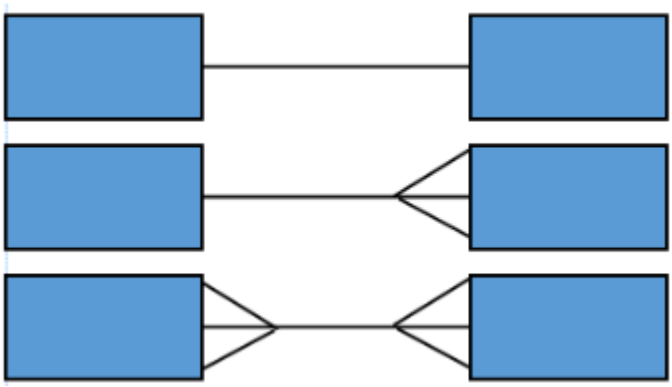
A primary key is a unique identifier for each row.

A composite primary key is a primary key made up of two fields. These two fields do not need to be unique on their own, but when combined, they must be unique.

A secondary key is another field of a table that is indexed and used to search it faster - e.g. searching by surname. They do not need to be unique for each record. However, the indexing table uses up extra storage space.

A foreign key is a field that contains the primary key of another table. These are used to draw relationships between tables.

Entity Relationship Diagrams



One to one is when one record of a table is related to exactly one record of another table.

One to many (or many to one if reversed) is when one record of a table can be related to multiple records of another table.

Many to many is when many records of a table can be related to many records of another table. This cannot be achieved without significant redundancy normally, so an intermediate linking table with composite primary keys from both tables is used to minimise data redundancy.

Section 5 - Networks and Web Technologies

Structure of the Internet and Internet Communication

The Internet

The internet is a network of networks set up to allow computers to communicate with each other globally. It is accessed via Internet Service Providers (ISPs), which usually provide routers for families and businesses to access the internet.

URLs

URLs are made up of:

- Method (http://, https://, etc)
- Host (www, mail, etc)
- Domain name (google, microsoft, etc)
- Second level domain (co, org, etc) (not always necessary) (there can be more levels, but this is uncommon)
- Top level domain (uk, ps, etc) (necessary)
- Location (e.g. webpage.html)
- Resource (e.g. #element)

> https://www.domainname.com/folder/subfolder/webpage.html#element

Domain names identify the areas or domains that internet resources reside in. Thees are structured into a hierarchy of smaller domains and written as strings separated by full stops as dictated by the rules of the Domain Name System (DNS).

Internet registrars hold records of all existing website names and details of domains that are available for sale. They resell domain names.

Internet registries own worldwide databases that hold records of all domain names currently issued to individuals and companies with their details. They also allocate IP addresses and keep track of which address(es) a domain name is associated with as part of the DNS.

Domain Name System

The Domain Name System (DNS) is a "hierarchical and distributed name service that provides a naming system for computers, services, and other resources on the Internet or other Internet Protocol networks" (source).

Domain names are made up of:

- A website name.
- Company second level domain (2LD) (not always necessary) (there can be more levels, but this is uncommon)
- Company top level domain (TLD).

To be a fully qualified domain name (FQDN), there must also be a host.

- Host (e.g. www, mail)

> https://www.bhf.org.uk
>
> https://www.bookstack.asadhussain.net

IP Addresses

An Internet Protocol (IP) address is a unique address that is assigned to a network device. It indicates where a packet of data is to be sent or has been sent from. Routers use the IP address to direct data packets accordingly.

Networks

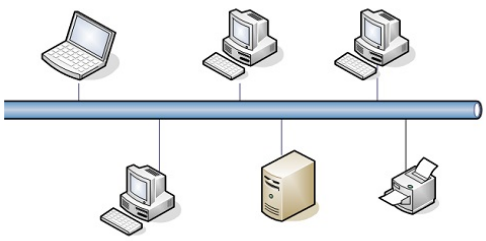
- A Local Area Network (LAN) consists of multiple devices connected over a small geographical area. Companies/individuals own and use their own hardware for a LAN, e.g. copper cables and ISP routers.
- A Wide Area Network (WAN) consists of multiple LANs connected over a large geographical area. Usually used by companies, this type of network usually requires renting infrastructure to connect LANs.

Network Topologies for LANs

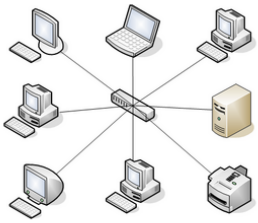
Topology	Description	Advantages	Disadvantages

(1) Physical bus	All devices are connected to a single cable.	Inexpensive to set up due to not a lot of cables required.	Single point of failure (main wire). With more traffic, performance degrades due to data collisions. Since every device has access to the cable, there is low security as all traffic can be seen.
(2) Star	All devices are connected to a central node, usually a switch, which routes data.	If one cable fails, other devices can still connect. Consistent performance, even under high load. No problems with data collisions. The system is more secure due to only one path between a device and the switch. Easy to add nodes to the network.	Costly due to more cables required. Single point of failure (central node).
(3) Partial/Full mesh	All devices are connected to each other, and access each other via these connections.	More reliable due to more interconnected nodes. - if one fails, another route can be taken via other nodes. With wireless, no cabling costs, so cheaper. Easy to add new nodes. Faster communication due to no need for a switch.	Physical implementations are expensive due to lots of cables.

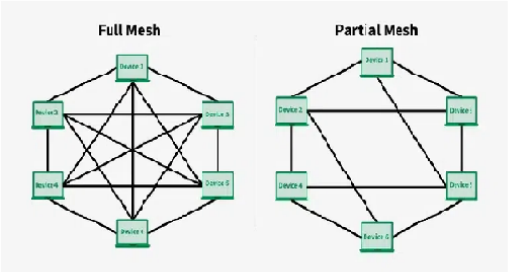
(1)



(2)



(3)



Network Security and Threats

Threats

Viruses

A virus is a piece of malicious software that has the ability to self-replicate. It exists in a host file, usually an executable file of some sort. It inserts its malicious code in other executable files to spread. A virus can be: spyware, which captures user data and sends them to a hacker; ransomware, which encrypts user data and demands payment for decryption.

A worm is another type of virus that doesn't require a host - it is self sustaining. It simply replicates itself and sends itself to other devices on a network or can hijack your email and send itself to your contacts.
E.g. the ILOVEYOU worm was a VBS script that destroyed every file on a user's computer, stole passwords, installed a backdoor, and used one's email to send itself to other people. The file was hidden as a TXT file named "LOVE-LETTER-FOR-YOU.TXT.vbs", but Windows hides file extensions by default, so users didn't know it was a script and expected the file to open in a text editor.

Trojans

A trojan presents itself as a legitimate file. Once run, it executes a payload, which may have different purposes based on its creator. The typical purpose of a trojan's payload is to install a backdoor in a computer, which is a route by which hackers can access a computer and perform more malicious tasks, such as spying on you or stealing your passwords.

System Vulnerabilities

Out-of-date software can have security flaws that can be exploited by hackers to gain access to a system. It thus makes it very important to keep software up-to-date.

It can also be just due to bad coding practices. An example of an exploit is SQL injection, where SQL statements are added to input boxes in a website, which can lead to the execution of these statements, allowing hackers to bypass login or just dump tables of data for the hacker to view.

Social Engineering

Social engineering involves the manipulation of people to gain unauthorised access to their/a system. This can be as trivial as peeking over their shoulder to see them type their password into their computer, or (as seen above in the ILOVEYOU worm) hiding the fact that a file is an executable script and making people think it's genuine enough to be opened.

An example is phishing, which is usually an email or website that impersonates a legitimate company like Google or Microsoft.

Pharming is the redirection of users to a fake copy of a trusted website, usually by alteration of DNS records or host files. This allows the collection of data like emails and passwords.

Data Interception

This is the unauthorised access of data while it is being transmitted over a network. If not encrypted, it can be read by a hacker. In a man-in-the-middle attack, the hacker is a proxy between the victim and the network, and can manipulate inbound and outbound data. E.g. the victim can be directed to a phishing login page for their email.

DoS and DDoS Attacks

DoS (denial of service) attacks involve sending massive numbers of requests to a server/network, causing bandwidth to be consumed, slowing down the network to the point of access to a provided service being denied.

DDoS (distributed denial of service) attacks involve multiple infected computers, usually infected from a virus, sending massive numbers of requests instead of just one. The increased number of requests makes DDoS attacks much more devastating than DoS attacks. A collection of infected computers used in DDoS attacks is called a botnet.

Security Measures

Firewalls

A firewall monitors and controls incoming and outgoing network traffic. It serves as a barrier between a computer and a network. It can be used to restrict access to specific applications and websites, controlling which resources can connect to or be accessed via the network. It can protect against:

- Hackers
- Malware
- DoS and DDoS attacks

A firewall uses packet filtering.

- Static (stateless) filtering involves examination of source and destination: IPs; protocols; and port numbers, all found in a packet header. Unauthorised requests are blocked based on a set of rules, which the above 3 pieces of information are compared with to authorise or reject requests.
- Dynamic filtering (stateful) involves the live analysis of packets going between two verified clients. Any detected anomalies are flagged based on temporary and adaptive rules for the session, and packets can be rejected or the connection can be dropped.

Proxy Server

A proxy server acts as an intermediary between a user's device and the internet, forwarding requests and returning responses on the user's behalf. This helps hide the user's IP address and allows organisations to monitor and control network traffic. It can protect against:

- Malware
- Hackers
- Phishing

A proxy server also maintains cached data for visited websites, speeding up website loading times. Network requests can also be filtered, so proxy servers are used in institutions like school to block access to blacklisted websites.

Encryption

Encryption scrambles data using a key to make it unreadable. Attackers cannot understand it if intercepted during transmission. It can protect against:

- Data interception
- Device/Data theft

Anti-Malware Software

Anti-malware software scans files and programs and compares them to a database of known malware signatures to identify them as malicious. It can quarantine, block, and remove suspicious files to prevent them from harming the system. They protect against malware such as:

- Viruses
- Worms
- Trojans
- Spyware
- Ransomware

Section 7 - Data Structures

Storing "lists" of data

A static data structure is fixed in size.

A dynamic data structure has a variable size. It does this with the use of a heap, which is a portion of memory from which space is automatically allocated and deallocated as required.

Arrays

Arrays are static data structures, where items are all of the same type and accessed via indices. A zero-indexed array means the first item is accessed with the index 0.

A two-dimensional (2D) array is where each item is a 1D array. This can be visualised as a table. Individual items are referred to as "array[row, column]" in OCR reference language. You can have more than two dimensions for an array.

Tuples

Tuples behave and are accessed like an array. The difference is that tuples are immutable, meaning their values cannot be changed, and tuples can hold data of more than one type.

Records

[Links to databases, strikingly similar to attributes in OOP]

Best for data storage. Each record has multiple fields, each of which hold one data type. They are defined, declared and accessed as follows in OCR reference language:

```
// Define
record1 = record
  field1
  field2
  field3
endrecord
```

```
// Declare
examplerecord : record1

// Access
examplerecord.field1 = "Hello World"
examplerecord.field2 = "Fizz"
examplerecord.field3 = "Buzz"
print(examplerecord.field1) // Hello World
```

Abstract data types

Data types created by the programmer rather than the language. A logical description of how the data is viewed and manipulated, but the user doesn't need to necessarily know *how* this is done. Good example of abstraction and encapsulation.

E.g. when you call "random.random()" in Python, you aren't concerned with how it works - just the output.

Lists

A list is a dynamic data structure that can contain an unlimited number of items which can be of different data types.

Methods:

```
isEmpty()           // Returns if the list is empty.
append(item)        // Adds a specified item to the end.
remove(item)        // Removes the first instance of the specified item.
search(item)        // Returns if the item is in the list or not.
length()            // Returns the length of the list.
index(item)         // Returns the index of the first occurrence of the item.
insert(pos, item)   // Inserts an item in a position, moving the already existing one after
it.
pop()               // Removes the last item in a list and returns it.
pop(pos)            // Removes an item by an index in a list and returns it.
```

Linked lists

A linked list is a dynamic data structure used to hold an ordered sequence

Queues

A first-in-first-out data structure (FIFO). New elements are added to the end, i.e. "enqueued", and items to be retrieved/removed are "dequeued".

Usage examples:

- Printer spooling.
- Task scheduling.
- Call centre systems.
- Handling website traffic.
- Characters typed on a keyboard.

Core methods:

```
isFull()         // Returns True or False depending on if the queue is full or not.
                 // True if the end pointer is equal to the length of the array.
                 // May not be needed if implemented with a list instead of an array.

isEmpty()        // Returns True or False depending on if the queue is empty or not.
                 // True if the end pointer is one less than the front pointer.
                 // (Not equal, because that indicates there is one item in the list)

enqueue(item)    // Adds an item to the end of the queue if it isn't full.

dequeue()        // Removes the first item in the queue if it isn't empty.
```

A queue is typically implemented with an array (or list), a front pointer, and an end pointer.

- The array ensures the queue only contains a fixed number of items of the same type.
- (A list may be used instead if the size of the queue cannot be determined beforehand.)
- The front pointer points to the first item in the queue. This is moved forward when "dequeue" is called.
- The end pointer points to the final item in a queue. This is moved forward when "enqueue" is called.

There are three types of queue:

Linear	• Can be implemented as a list or an array. • Pointers move forward. • As elements are dequeued, space at the front of the queue is never accessed again, so memory is wasted.

Circular	• Can be implemented as an array. • Pointers move forward, but circle back to the front of the array when the end is reached. • Fixes the issue of wasted memory. • Doesn't offer the flexibility of a dynamic data structure - best for queues in which the size can be predetermined. • More complicated for the programmer to implement.
Priority	• The logical order of items is determined by their priority, with higher priority items at the front. • It is possible for higher priority items to be enqueued to the front of the queue or inserted in between items.

Linked lists

Section 10 - Computational Thinking

Abstraction

Thinking Ahead

Thinking Procedurally

Thinking Logically & Concurrently

Backtracking

Backtracking is when different sequences of actions are tried to obtain a solution. If one path doesn't work, you "backtrack" to the last known working point.

An example of this is depth-first traversal of a graph/tree, where you go down one branch fully before returning to the last node to check for any unchecked branches that may be coming out of it.

Data Mining

Data mining involves the processing of large data sets to find patterns and relationships. This enables one to produce insights on these large data sets.

An example of this is in social media algorithms, which personalise your feed based on your interests to keep you on the platform for longer.

Heuristics

A heuristic approach describes a "good enough" approach to a problem that does not necessarily produce a perfect solution, but minimises program (time and/or space) complexity.

An example is the A* algorithm vs Djikstra's algorithm for pathfinding.

Performance Modelling

Pipelining

Visualisation

Parallel Processing

Concurrent Processing