*This is a **bite-sized** summary of theory for your GCSE Computing exam.*

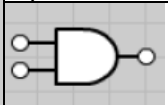
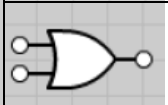
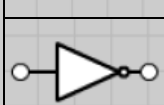
*It is the **bare-minimum** you MUST be able to remember.*

*To ensure you get that "A grade", read over the **complete** course text book.*

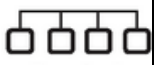
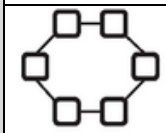
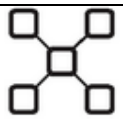
Section 1 : Computer Systems

CPU **Fetch Decode Execute** **(Clock Cycle)**	**CPU** Fetches instructions from RAM – **Decode** part examines the binary instruction to see which part is 'data' (the 'operand') and which is the 'instruction itself' (what you actually *do* with the data – the 'op-code') **Execute** is when the **ALU** performs Arithmetic and/or Logic calculation on the data. This happens in a **clock cycle**.
Controller **Registers** **Buses**	The **Controller** makes sure data is processed in the correct order. Data waiting to be processed is stored in temporary **Registers**. The **Address Bus** gets the next piece of needed from the registers and RAM, whilst the **Data Bus** receives the inputs from other parts of the computer and sends processed data out.
Clock Speed	Clock speed measured in **Hertz**. The amount of Instructions processed (**F**etch/**D**ecode/**E**xecutes) per second. 3GHz = (3 Giga Hertz / 3 *Billion* cycles)
Cache Memory	**Cache** is inside of the CPU. Remembers **frequently processed** instructions / data. Saves having to read from RAM all the time. Cache is fast but expensive.
Number of Cores	Dual (2) and Quad (4) processors can perform more calculations **simultaneously**.
Types of Processor	**Reduced** Instruction Set Computer (**RISC**) (Perform simpler, broken-down instructions – **less** circuitry and heat – ideal for small devices – e.g phones) **Complex** Instruction Set Computer (**CISC**) – What a traditional CPU does.
RAM / ROM and Flash Memory	**RAM** (Random Access Memory) Holds working temporary data, operating system and running programs. **ROM** (Read Only Memory) Holds data, programs etc permanently. **Flash** is special **RAM** memory that is **Non-volatile** (keeps data without power) – Used in pendrives / SD-Cards. 'Solid-State' – no moving parts.
Storage Media: Suitability, Capacity, Durability, Portability, Speed	**Flash drive** (Everyday use / 2GB – 64GB / Very Durable / Very Fast) **External hard drive** (Home back-ups / 1-4TB / Not Very Durable / Fast) **CD/DVD** (Multimedia / CD: 700MB - DVD 4GB / Reasonably Durable / Slow) **Magnetic tape**(Industrial archives / 1-4TB / Reasonably Durable / Very Slow)
Graphical User Interface (GUI)	Novice **friendly**, (mostly) language independent, Also known as **WIMP** (Windows, Icons, Menus, Pointer) – Easy to learn, but uses a lot of RAM Memory.
Menu Interface	Commands broken down into **categories** (e.g Edit, View). Restricts.
Command Line	Non-GUI **Text only** interface for expert (technical) users. Less memory. **Quicker** to perform some tasks, **more powerful** features but commands most be **known**.

Section 2 : Data Representation

Binary Number System	A **base-2** number system that represents the **ON** and **OFF** states of an electrical circuit.
Character Sets	**ASCII**: A table of 8 bit Binary patterns and a corresponding character it represents. Limited to 128 characters only (main English keyboard) **Unicode**: Uses 16 bits to have thousands more characters (international)
Logic Gates	Logic Gates – Represent the **flow of electricity** through a CPU circuit and how it is 'channelled' to make decisions. **Truth Tables** are used to represent all the different combinations of '1' and '0' input and what the output would be.
	 **AND** Gate: Needs **both** inputs to be '1' to produce '1' as an output
	 **OR** Gate**:** Only needs one of **either** input to be '1' to produce '1' as an output
	 **NOT** Gate: **Reverses** the input: '1' will produce '0' as output ('0' in = '1' out)
Truth Table	Shows what the output will be for every possible **combination** of input.
Units of Storage	A **Bit** (1 or a 0) / **Byte** (8 bits) / **Kilobyte** (1024 bytes) / **Megabyte** (1024 Kb) / **Gigabyte** (1024 MB) / **Terabyte** (1024 GB) / **Petabyte** (1024 TB)
Data Types	**Integer** (Whole Number) **Real** (Decimal) **Boolean** (YES / NO Value) **Character** (Any alpha-numeric character) **String** (Sentences)

<table>
<tr><td>Instructions & Data</td><td colspan="2">First 4 bits of a byte are the instruction part – the 'op-code' (e.g ADD, LOAD, STORE etc) the rest is the data itself - the 'operand' (the part that needs to be examined and calculated)</td></tr>
<tr><td>Sound</td><td colspan="2">Real life analogue sounds are sampled (captured) and converted into a sequence of binary digits. More sample 'points' = a more lifelike the sound (but larger size)</td></tr>
<tr><td>Pixels</td><td colspan="2">A Pixel ('picture element') contains bit-sequence to represent a single colour – The more bits greater the bit-depth meaning more colour can be represented in an image – becomes more life-like. (but larger file size) Bit-depth of 1 represents two colours (e.g Black and White – Binary 1 and 0) / Bit-depth of 2 represents four colours (Binary 00, 01, 10, 11) etc…</td></tr>
<tr><td>Metadata</td><td colspan="2">Data 'about' data – Metadata for an image file is details about the height / width / byte-size / date etc of the image. Without it image would be distorted.</td></tr>
<tr><td>Binary Conversion</td><td><table><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td>4</td><td>2</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>Would give deanery 70</td><td>Working out in reverse (e.g "41 in Binary") Start with next lowest number on scale (e.g 32) remainder 9 (which is made up from a 8 and a 1) = Therefore, 41 = 00101001</td></tr>
<tr><td>Hex number system</td><td colspan="2">Base 16 counting system. Digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 are used to represent 1–9 and then the characters A, B, C, D, E and F are used to represent 10 –15.</td></tr>
<tr><td>Hex to Denary:</td><td>First, convert the LEFT element into a number. If A-F then multiply by 16, (Remembering A= 10, B = 11 etc..) Then repeat with the RIGHT element, then add the two numbers.</td><td>EA
E=14
14*16 = 224
A = 10
224+10 = 234 (EA hex is 234 Denary)</td></tr>
<tr><td>Denary to Hex:</td><td>Divide the number by 16. Then take the whole number (before the decimal) and place as the LEFT element. If 10-15 then convert to the correct letter.
If there is a remainder, multiply this by 16 and this is the RIGHT element, if 10-15 convert to the correct letter.</td><td>210
210/16 = 13.125
13=D
0.125*16 = 2
D2 (210 Denary is D2 hex)</td></tr>
<tr><td colspan="3"><u>Section 3 : Computer Systems</u></td></tr>
<tr><td>Functions of an Operating System</td><td colspan="2">Provides a User Interface: GUI or Command Line to allow system to be used.
Memory Management: Allocates working RAM memory to programs as needed. May prioritise some tasks over others (allows some to work in 'background')
Hardware Management: Allows system to interact with peripheral devices using drivers (software that acts as a translator between the different manufacturers devices) OS also allows a file storage system. OS also provides different levels of user access to keep files and folders secure (e.g. Deny Access, Read Only Access, Write Access and Full Access)</td></tr>
<tr><td>Software Libraries</td><td colspan="2">Common functions owned by the OS and shared by other programs (e.g saving, printing, searching etc…) Saves individual programs duplicating the features.</td></tr>
<tr><td>Software Integrated Development Environments</td><td colspan="2">Allows users to write computer programs. Run / test out code to identify errors. Preview the results of the code in different outputs (e.g how a webpage would look in different resolutions, or a phone app when installed on different phones) Python's IDLE and Greenfoot are Software Development Environments.</td></tr>
<tr><td>Disk Formatting</td><td colspan="2">Disk 'contents' (File Allocation Table) is wiped clean. Prepares disk on first use and gives the impression disk is empty.</td></tr>
<tr><td>Disk Compression</td><td colspan="2">File is 'zipped' (space removed, complex algorithms performed to re-structure data) so that the file size is reduced. Can be unzipped with no data loss.</td></tr>
<tr><td>Disk Defragmentation</td><td colspan="2">As files are used and re-saved over time they become scattered and saved across different parts of the physical disk, slowing down access to them. Defragmenting will reorganise the will by putting pieces of related data back together again.</td></tr>
<tr><td>System Restore</td><td colspan="2">Restore (roll-back) system settings that are accidentally / maliciously changed.</td></tr>
<tr><td>Firewall</td><td colspan="2">Filters incoming and outgoing network traffic to and from a computer. Stops external hacks in and filters access to 'inappropriate websites' going out.</td></tr>
<tr><td>Common Applications</td><td colspan="2">Avoid Microsoft brand names in your exam and use the following 'generic' phrases: "Word Processing Software" (instead of… Word) "Spreadsheets" (Excel) "Database Software" (Access) "Presentations Software" (Powerpoint) "Graphics" (Photoshop)</td></tr>
<tr><td>Common Utilities</td><td colspan="2">Utilities do specific (non-creative) functions. Usually to keep the system running correctly (Anti-Virus, Disk-Cleaners, File Convertors, Download Helpers, etc) An OS will have lots of these accessible from the Control Panel</td></tr>
</table>

Section 4 : Networks		
Network Hardware	**Router / Modem**: Connects LAN to WAN / Converts analogue to digital **Hub**: Shares one signal with many devices (Easy way to add new pcs to one port) **Switch**: Sends specific data from one item to another specific item (used to connect many users to a single server) **Server**: Controls network – allows log-on – stores files centrally **Repeater**: Allows network to span large distances – repeats signal down cable Software **Port**: A setting on a server or firewall to grant or deny access	
Network Topologies Consider: *Different Ways To Set-Up A Network* – Speed, Adaptability, Security And Cost (All considerations in deciding on which to use.)		**BUS**: *Data flows along main backbone cable* (terminators at ends) Easy to add new workstations / Cheaper - Less Cable If problem with main backbone – Network Stops More workstations = slower speeds. Only one PC at a time can transmit data down it.
		**RING**: *Data 'token' passes from one PC to the next* No reliance on central PC. No Data collisions as data travels in one direction only. Network needs to be shut-down to add more devices. If one cable breaks – whole network fails. Cheap – less cable.
		**STAR**: *Each device has a cable direct to the main server.* Reliable – If one cable fails, other users not effected. Easy to add new pcs. Expensive as uses most cable. Needs central server to work.
Protocols	A protocol is a set of **rules** that describe how data is to be transmitted across the network. E.g. When the communication will start and end / The transmission speed Error checking procedures (i.e was data sent the same as received)	
TCP / IP Protocol	(**T**ransmission **C**ontrol **P**rotocol and **I**nternet **P**rotocol) Used to **exchange data** between pcs on a network and **route packets** between networks / the Internet.	
HTTP Protocol	(**H**yper**T**ext **T**ransfer **P**rotocol) Used on WWW to transfer **webpages** and web content from the website host server to the computer requesting the page.	
FTP Protocols	(**F**ile **T**ransfer **P**rotocol) Used to transfer files between computers over a network.	
SMTP Protocols	(**S**imple **M**ail **T**ransport **P**rotocol) Used to transport and store emails.	
Data Packets (Three parts to a packet of data transmitted across the internet)	(**Part 1**- Packet Header): Contains the **destination** ip **address** (or MAC address if on LAN) and details about the length of the data expected to be sent. (**Part 2**) **Actual Data** itself: Located between header and footer. (**Part 3**) Packet Footer: Data to state this is the **end** of the packet and also **errror-checknig** to ensure all the data **sent** has been **received** fully.	
Network Security	**Passwords** should be a combination of letters and numbers. User accounts and permissions set to deny or grant access. Firewalls and physical security also used.	
Encryption	**Converting** data to be sent over a network into something that is **unreadable**. Data is first **encrypted**, then sent over the network. Then receiver decrypts.	
Network Policies	**Acceptable Use Policy** – Set of Rules / outlines behaviour expected (do not install software, bully on-line etc) All users have to sign before using network.	
Section 5: Internet & Communication		
Common file types found on internet.	Most of these can **compress** / stream data. (HTML – Make pages / CSS – Format page styles / JPEG – compressed image / mp3 – compressed audio /swf – flash video (interactive multimedia)	
Lossly Compression	**Reduces** file size by **physically removing** data. Image compression reduced the amount of colours used. Audio compression removed high / low end frequencies.	
Lossless Compression	Files are compressed with **no data loss**. Algorithm that looks for 'patterns' in the data that can be removed. (e.g white space) or re-organising data. (e.g *Zipping*)	
IP Address	Identifies a **point of connection** to the internet (shared home / business router)	
MAC Address	Identifies a **physical item** on network (like serial number /**M**edia **A**ccess **C**ontrol)	
DNS Server	**Looks up** a www… address and directs traffic to the actual IP address of website. Allows for users to type in web addresses rather than complicated IP numbers.	
ISP	**I**nternet **S**ervice **P**rovider (Provides connection to wider internet – e.g Virgin / BT)	
Cookies	Small text file created when **visiting a website**. Stores info about the time / date of the visit. Used to **monitor browsing habits** and generate targeted **adverts**.	
Search Engines	Send **bots** (programs) out to **scan** all webpages in existence. They report back site / page contents to compile a huge **database** of what the internet consists of. This database is searched when the user types **search criteria** into engine. Google uses a complicated **algorithm** to work out what it considers the **most relevant** websites on a particular topic and **ranks** them high in it's database.	

Section 7: Programming	
Variables	**Dynamic** values that **change** throughout a program. It may have an **initial** value, but will then change **automatically** – from either a calculation or user input. *Examples*: Score, Age etc… **Local** Variables are declared within a **function** and only used within that function. **Global** Variables are used **throughout** program.
Constant	**Static** value that **remains the same** throughout a program. *Examples*: Pi = 3.14, VAT = 17%, Age Restriction = 18 etc…
Array	Storing values (of the same type) in a list. E.g: Scores = [10,15,8,10] The first position in an array is [0]. In this example Scores[2] would be *8*.
Data Types	**Number** (Whole numbers 0-9) / **Integer** (Decimals) / **Character** (Letters, $ym8ols and numb3rs) / **String** (Sentence with spaces) / **Boolean** (Yes / No)
Logical Operators	**==** (Is equal to / Matches) **!=** (Not equal to) **>** (More Than) **<** (Less Than) **>=** (Greater than or equal to) **<=** (Less than or equal to)
Iteration (Looping)	**FOR** Loop: Runs a **specific number of times**, for example: FOR number IN things to revise OUTPUT "Revise this thing" **WHILE** Loop: Runs **until a condition has been met**, for example: WHILE grade != "A*": OUTPUT "Keep Revising"
High Level Languages	These allow programmers to type using **recognisable** 'English' **keywords** and **syntax**. E.g: **print (" ")** They are **easier** to learn, understand and write in than low-level languages. Python and Java are examples.
Low Level Languages	**Machine Code**. This does not contain any recognisable words. Made up from hexadecimal bit sequences. It is understood by the CPU.
Translators (Interpreter)	**Converts** high level language **one line at a time** into machine code (so that CPU can read it). **F5** in python is an does this. Will **stop** when an error occurs.
Translators (Compiler)	**Bundles** all of the high-level code into a single **executable file**. (Greenfoot does this and will only 'run' when there are **no errors** in the code.
Assembler	**Assembly language**. Uses **abbreviated** words (**mnemonics**) (2 or 3 letter codes) Easier to remember than machine code. (E.g: *LDA* = LOAD, *STA* = STORE) The **Assembler** coverts assembly language into machine code. (Breaking the binary byte into an **opcode** (instruction) and an **operand** (actual data to use)
Programming Errorz	**Syntax** (Typing errors / misspellings etc) / **Run time**/execution (Crashes when running – attempts something the computer cannot do) / **Logical** (Program runs, but gives **wrong results**) / **Linking** (Calling a OS library that does not exist) **Rounding / Truncation** (Errors produced when rounding numbers up or down)
Section 9: Ethical, Legal Aspects	
Data Protection Act	**Data Protection Act:** States what companies can and cannot do with people's personal data. Person is known as '**Data Subject'** – has the right to see what data company holds about them. (Police and Military may be except from showing) Law states Company must keep data **secure** / up-to-date / accurate and obtain information fairly / Prevents selling data onto other companies (**junk-mail**)
Computer Misuse Act	Making accessing (**hacking**) another computer **without permission** an offence. Penalties for (1) Attempted Access (2) Access (3) Modifying files (4) Deleting Files.
Code of Conduct	Defines **acceptable behaviour** within an organisation. E.g: Don't give passwords out / Don't install programs / Don't send offensive emails / No porn whilst working.

Example Pseudocode (Use CAPS for programming procedures / Indent where needed)

```
DEFINE (MainProgram)

DECLARE VarLimit = 10 (integer)
DECLARE NumbersArray = [1,2,3,4]

OUTPUT "Input a new number"

INPUT: NewNumber (integer)

IF NewNumber > VarLimit THEN
       OUTPUT "Number too big"
ELSE OUTPUT "Number accepted"
       NumbersArray = [NumbersArray + NewNumber]

END (MainProgram)
```

```
DEFINE (MainProgram)

DECLARE Count = 0 (integer)

SET Full = FALSE (boolean)

WHILE Count < 10 THEN:
       OUTPUT "Input Something"
       SET Count = Count + 1
       INPUT NewStuff (string)

OUTPUT "Limit Reached"
SET Full = TRUE

END (MainProgram)
```