

|              |                                |                             |
|--------------|--------------------------------|-----------------------------|
| <b>Topic</b> | <b>1.1 Data Representation</b> | <b>1.1.1 Binary Systems</b> |
|--------------|--------------------------------|-----------------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| recognise the use of binary numbers in computer systems   |
| convert positive denary integers into binary and positive binary integers into denary (a maximum of 16 bits will be used) |
| show understanding of the concept of a byte and how the byte is used to measure memory size                               |
| use binary in computer registers for a given application (such as in robotics, digital instruments and counting systems)  |

| <b>Core Language</b>                                     |   |  |
|--|---|--|
| <b>Bit:</b> the smallest unit in binary                  | <b>Nibble:</b> half a byte  | <b>Byte:</b> 8 bits  |
| <b>Kilobyte:</b> 1024 bytes                              | <b>Megabyte:</b> 1024 kilobytes   | <b>Gigabyte:</b> 1024 megabytes  |
| <b>Terabyte:</b> 1024 gigabytes                          | <b>Denary:</b> "base 10"number  | <b>Register:</b> a small amount of memory inside the processor that stores information for the processor for immediate use |
| <b>Memory:</b> location for holding data and instruction | <b>Capacity:</b> the amount of information that is able to store in something | <b>Units:</b><br>byte-kilobyte-megabyte-gigabyte-terabyte-petabyte-zettabyte-yottabyte                                     |

|       |                         |                   |
|-------|-------------------------|-------------------|
| Topic | 1.1 Data Representation | 1.1.2 Hexadecimal |
|-------|-------------------------|-------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| represent positive numbers in hexadecimal notation  |
| show understanding of the reasons for choosing hexadecimal notation to represent numbers  |
| convert positive hexadecimal integers to and from denary (a maximum of four hexadecimal digits will be required)  |
| convert positive hexadecimal integers to and from binary (a maximum of 16 bit binary numbers will be required)  |
| represent numbers stored in registers and main memory as hexadecimal  |
| identify current uses of hexadecimal numbers in computing, such as defining colours in Hypertext Markup Language (HTML), Media Access Control (MAC) addresses, assembly languages and machine code, debugging [ hexadecimal is easier to write compared to binary ] |

| Core Language  |  |  |
|--|--|--|
| <b>Hexadecimal:</b> “base 16” number   | <b>MAC Address:</b> a hardware identification number that uniquely identifies each device on a network | <b>Assembly Language:</b> a low-level programming language   |
| <b>Machine Code:</b> a computer program written in machine language  | <b>Debugging:</b> the process of finding and resolving an error  | <b>HTML (HyperText Markup Language) :</b> the standard markup language for documents to display in a web browser |
| <b>Character Set:</b> the composite number of different characters that are being used and supported by a computer software and hardware | <b>Concatenate:</b> countable and uncountable of computer data   |  |

| Topic | 1.1 Data Representation | 1.1.3 Data Storage |
|-------|-------------------------|--------------------|
|-------|-------------------------|--------------------|

| Syllabus Learning Outcomes. You should be able to:  |
|---|
| show understanding that sound (music), pictures, video, text and numbers are stored in different formats  |
| identify and describe methods of error detection and correction, such as parity checks, check digits, checksums and Automatic Repeat reQuests (ARQ) |
| show understanding of the concept of Musical Instrument Digital Interface (MIDI) files, JPEG files, MP3 and MP4 files                               |
| show understanding of the principles of data compression (lossless and lossy) applied to music/video, photos and text files                         |

| Core Language  |  |   |
|--|--|---|
| <b>ASCII:</b> a standard code ( 7 bits for every character) used so that data can be moved between computers use different programs  | <b>Extended ASCII:</b> 128 new features added to the ASCII ( 8 bits for every character)   | <b>Unicode:</b> a computer standard for the consistent encoding ( 16bits)   |
| <b>MPEG:</b> a commonly used method for audio and video  | <b>JPEG:</b> a commonly used method for digital images (compressed bitmap)   | <b>AVI:</b> a commonly used method for audio and video  |
| <b>Bitmap:</b> a way in which an image is stored with a fixed number of bits (= units of information) for each unit of the image   | <b>Vector Image:</b> different from bitmap,defined in terms of 2D points   | <b>File Type:</b> a standard way that information is encoded for storage in a computer file                                 |
| <b>Mp3:</b> a standard way to store audio  | <b>RAW:</b> primary data   | <b>Lossless Compression:</b> opposite from lossy compression  |
| <b>Lossy Compression:</b> a data encoding method that compresses data by discarding (losing) some of it. The procedure aims to minimize the amount of data (for example: MPEG,JPEG,AVI)  | <b>MIDI:</b> Musical Instrument Digital Interface that is a technical standard description of instrument                         | <b>Error Detection:</b> detect errors that occur in the transmitter to the receiver   |
| <b>Parity Check:</b> a check made of computer data to ensure that the total number of bits of value 1 (or 0) in each unit of information remains odd or even after transfer between a peripheral device and the memory or vice versa | <b>Check Digit:</b> the last digit of a sequence used to check all other digits are correct and in the correct order             | <b>Checksum:</b> the total of the numbers in a piece of digital data, used to check that the data is correct                |
| <b>ARQ:</b> an error-control method for data transmission that always send the same thing to the receiver until it is received correctly   | <b>Colour Depth:</b> the color information stored in an image,the higher the bit depth of an image, the more colors it can store | <b>Resolution:</b> the quality of image (how many pixels in a certain area) produced by a printer or displayed on a monitor |

|   |  |   |
|---|--|---|
| <b>Bit Depth:</b> also known as colour depth                                  | <b>Sample Rate:</b> the rate at which samples of an analog signal are taken in order to be converted into digital form | <b>Sample Interval:</b> the distance or time between two measurements         |
| <b>Analog Data:</b> continuous real world data that a computer cannot process | ADC analog-to-digital converter➡<br>⬅DAC digital-to-analog converter   | <b>Digital Data:</b> none continuous data that can be processed by a computer |

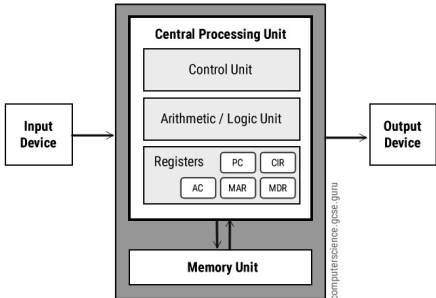

|       |                           |                   |
|-------|---------------------------|-------------------|
| Topic | 1.3 Hardware and Software | 1.3.1 Logic gates |
|-------|---------------------------|-------------------|

| Syllabus Learning Outcomes. You should be able to:   |
|--|
| use logic gates to create electronic circuits  |
| understand and define the functions of NOT, AND, OR, NAND, NOR and XOR (EOR) gates, including the binary output produced from all the possible binary inputs (all gates, except the NOT gate, will have 2 inputs only) |
| draw truth tables and recognise a logic gate from its truth table  |
| recognise and use the standard symbols used to represent the following logic gates:<br>NOT, AND, OR, NAND, NOR, XOR  |
| produce truth tables for given logic circuits  |
| produce a logic circuit to solve a given problem or to implement a given written logic statement   |

| Core Language  |   |  |
|--|---|--|
| <b>Truth Table:</b> a mathematical table that shows outputs from all possible inputs from a Boolean function         | <b>Boolean:</b> connected with a system, used in computing and electronics, that uses only the numbers 1 (to show sth. is true) and 0 (to show sth. is false) | <b>Logic Circuit:</b> a collection of logic gates for performing logical operations on input signals |
| <b>Operator Precedence:</b> the order of operation that presented in a mathematical expression                       | <b>AND:</b> gives positive result (1) when two inputs are positive  | <b>OR:</b> gives positive result when 1 or more inputs are positive                                  |
| <b>NOT:</b> gives positive result when the only input is negative (0) or give negative result when input is positive | <b>NOR:</b> "not or gate"<br><b>NAND:</b> "not and gate"  | <b>XOR:</b> gives positive result when only 1 input is positive                                      |

|       |                           |   |
|-------|---------------------------|---|
| Topic | 1.3 Hardware and Software | 1.3.2 Computer architecture and the fetch-execute cycle |
|-------|---------------------------|---|

|  |
|--|
| <b>Syllabus Learning Outcomes. You should be able to:</b>  |
| show understanding of the basic Von Neumann model for a computer system and the stored program concept (program instructions and data are stored in main memory and instructions are fetched and executed one after another) |
| describe the stages of the fetch-execute cycle, including the use of registers and buses   |

| Core Language  |  |  |
|--|--|--|
| <p><b>Von Neumann Architecture:</b> consists of the CPU and the memory unit, and Inputs/outputs, has the ability to input, process, output and store (IPOS)</p>   | <p><b>CPU:</b> stands for central processing unit, an electronic circuitry that does computer programs (CPU=Processor + RAM)</p> <p>  Video<br/> <a href="https://drive.google.com/file/d/1mT1cijzMW9cRRoDvt2h6R-bK6rrXcgDv/view">https://drive.google.com/file/d/1mT1cijzMW9cRRoDvt2h6R-bK6rrXcgDv/view</a> </p> | <p><b>RAM(volatile):</b> random-access memory, as its name it can be read and changed in any order and commonly used to hold data and instructions that the computer is currently processing</p> |
| <p><b>Processor:</b> the chip that carries out all processing within a computer system</p>   | <p><b>Fetch-decode-execute cycle:</b> the cycle which the CPU follows once the computer is turned on, there are 3 stages, fetch, decode (instruction) and execute</p>  | <p><b>Writing:</b> typing text, outputting data</p>  |
| <p><b>Reading:</b> retrieval of data and instructions from main memory so that it can be processed</p>   | <p><b>Memory Address:</b> represent a memory location within the main memory<br/>Often expressed in hexadecimal</p>  | <p><b>Volatile:</b> temporary, and once the system is shut down, the data will be removed (faster than non-volatile memory)</p>  |
| <p><b>Non-volatile:</b> data is kept completely all over the time (safer than volatile memory)</p>   | <p><b>ROM(non-volatile):</b> read-only memory, holds the boot-up instructions for the computer system</p>  | <p><b>Cache:</b> a superfast memory within the CPU ( made of registers )</p>   |
| <p><b>Main Memory (primary memory):</b> RAM and ROM</p>  | <p><b>Link:</b><a href="https://www.computerscience.gcse.guru/theory/von-neumann-architecture">https://www.computerscience.gcse.guru/theory/von-neumann-architecture</a></p>   | <p><b>Secondary Storage :</b> external storage and use for storing files for long terms created by users</p>   |
| <p><b>Bus:</b> the connection between components inside a computer<br/>Control bus(carries control signal within the processor, bidirectional);<br/>Address bus(carries the address to the RAM of the next location to be accessed, monodirectional);<br/>Data bus(carries the data that is going to be processed)</p> | <p><b>Virtual Memory:</b> a part of the secondary storage (hard drive)which can be used temporarily as a RAM but it is slower</p>  | <p><b>Stored Program Concept:</b></p>  |

|  |   |   |
|--|---|---|
| <b>Instructions:</b> single operations of the processor  | <b>Clock Speed:</b> how many times in one sec a computer can execute its instruction, demonstrate the speed of the processor<br>A computer can overclock its processor but this may damage. | <b>Register:</b> a memory location within the CPU that holds a very small set of data (instructions, storage address etc.) for a short period of time |
| <b>MAR:</b> memory address register, holds the next memory location that the process need to access<br>Do not use acronyms in exams! | <b>MDR:</b> memory data register, holds the contents found at the address held in the MAR<br>Do not use acronyms in exams!  | <b>CIR:</b> current instruction register, holds the current instruction that is currently being processed<br>Do not use acronyms in exams!            |
| <b>AC:</b> accumulator, where intermediate arithmetic and logic results are stored<br>Do not use acronyms in exams!                  | <b>PC:</b> program counter, contains the location of the next instruction to be accessed<br>Do not use acronyms in exams!   |   |

|       |                           |                     |
|-------|---------------------------|---------------------|
| Topic | 1.3 Hardware and Software | 1.3.3 Input Devices |
|-------|---------------------------|---------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| describe the principles of operation (how each device works) of these input devices: 2D and 3D scanners, barcode readers, Quick Response (QR) code readers, digital cameras, keyboards, mice, touch screens, interactive whiteboards, microphones |
| describe how these principles are applied to real-life scenarios, for example: scanning of passports at airports, barcode readers at supermarket checkouts, and touch screens on mobile devices   |
| describe how a range of sensors can be used to input data into a computer system, including light, temperature, magnetic field, gas, pressure, moisture, humidity, pH and motion  |
| describe how these sensors are used in real-life scenarios, for example: street lights, security devices, pollution control, games, and household and industrial applications   |

| Core Language  |  |   |
|--|--|---|
| <b>Peripheral:</b> an external device connected to a computer system, it can be both wired or wireless | <b>Input Device:</b> convert analog data into digital data                                   | <b>Output Device:</b> convert digital data into analog data                                 |
| <b>Sensor:</b> detect some kinds of physical change in the environment                                 | <b>Actuator:</b> motors that are commonly in conjunction with sensors to control a mechanism | <b>Feedback Loop:</b> a circuit that feeds back some of the output to the input of a system |



|       |                           |                      |
|-------|---------------------------|----------------------|
| Topic | 1.3 Hardware and Software | 1.3.4 Output Devices |
|-------|---------------------------|----------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| describe the principles of operation of the following output devices: inkjet, laser and 3D printers; 2D and 3D cutters; speakers and headphones; actuators; flat-panel display screens, such as Liquid Crystal Display (LCD) and Light-Emitting Diodes (LED) display; LCD projectors and Digital Light Projectors (DLP) |
| describe how these principles are applied to real-life scenarios, for example: printing single items on demand or in large volumes; use of small screens on mobile devices  |

| Core Language   |  |   |
|---|--|---|
| <b>Peripheral:</b> an external device connected to a computer system via a port (both wired and wireless) | <b>Input Device:</b> sends information to the computer to process                                  | <b>Output Device:</b> display the scene on the computer or follow the instructions of the computer to do work |
| <b>Sensor:</b> an input device which record data from the real world to send to the computer              | <b>Actuator:</b> an output device which generate a real world movement after receiving instruction | <b>Feedback Loop:</b> the process of using the incoming information, and reacting to it                       |

|       |                           |   |
|-------|---------------------------|---|
| Topic | 1.3 Hardware and Software | 1.3.5 Memory, storage devices and media |
|-------|---------------------------|---|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| <p>show understanding of the difference between: primary, secondary storage; off-line storage and cloud. Provide examples of each, such as:</p> <ul style="list-style-type: none"> <li>Primary(holding data): Read Only Memory (ROM) and Random Access Memory (RAM)<br/>ROM-solid-state, non-volatile, boot-up files<br/>RAM-solid-state, volatile</li> <li>Secondary(storing data): hard disk drive (HDD) and Solid State Drive (SSD);</li> <li>Off-line (can remove from computer and take away) : Digital Versatile Disc (DVD), Compact Disc (CD), Blu-ray disc, USB flash memory and removable HDD</li> <li>Cloud: store data online in a server</li> </ul> |
| describe the principles of operation of a range of types of storage device and media including magnetic, optical and solid state  |
| describe how these principles are applied to currently available storage solutions, such as SSDs, HDDs, USB flash memory, DVDs, CDs and Blu-ray discs   |
| <p>Calculate the storage requirement of an image:</p> <p>Storage= (resolution*bit depth) / (8* 1000<sup>n</sup>) ← if bit depth is given in bits<br/> Storage= (resolution*bit depth) / (1000<sup>n</sup>) ← if bit depth is given in bytes</p> <p>Calculate the storage requirement of a sound file:</p> <p>Storage=(duration*sample rate*bit depth*channels) / (8*1000<sup>n</sup>) ← bit depth in bits<br/> Storage=(duration*sample rate*bit depth*channels) / (1000<sup>n</sup>) ← bit depth in bytes</p> <p>⚠ channels can be: mono(*1), stereo(*2) or quad(*4)</p>   |

| Core Language   |    |   |                   |  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
|---|----|---|-------------------|--|--------------------|--|--|--|--|---------|--|--|--------|--|-------|--|--------|-------|-----|-------|------|----|----------|------|-----|----------|-------------------|----|----------|-------------------|-----|----------|-------------------|----|----------|-------------------|-----|----------|-------------------|----|----------|-------------------|-----|----------|-------------------|----|----------|-------------------|-----|----------|-------------------|----|---------|-------------------|-----|----------|
| <b>Solid State Storage:</b> they store data in electric circuits made of transistors, charged circuits represent 1 and non-charged circuits are 0s.<br><br>Examples: SSDs, SD cards, USB flash drives and RAM |    | <b>Optical Storage:</b> a storage device which requires light to store and read the data.Data is written onto the optical storage disc by a laser and is read from the disc by another laser.<br><br>Examples: CD, DVD  |                   | <b>Magnetic Storage:</b> use magnetic fields to magnetise tiny individual sections of a metal spinning disk to store data. Magnetised sections represent 1s and demagnetised sections represent 0s.<br><br>Examples: floppy disks, hard disks(HDD) |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| <b>Capacity:</b> the maximum amount of data stored in any storage device  |    | <b>Bit:</b> the smallest unit in binary, representing a 1 or 0  |                   | <b>Nibble:</b> 4 bits & half of a byte   |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| <b>Kilobyte:</b> 1000 bytes   |    | <b>Megabyte:</b> 1000 kilobytes   |                   | <b>Gigabyte:</b> 1000 megabytes  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| <b>Kibibyte:</b> 1024 bytes   |    | <b>Mebibyte:</b> 1024 kibibytes   |                   | <b>Gibibyte:</b> 1024 mebibytes  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| <b>1000<sup>2</sup> vs 1024<sup>2</sup>:</b>  |    | <table><tr><th colspan="5">Multiples of bytes</th></tr><tr><th colspan="3">Decimal</th><th colspan="2">Binary</th></tr><tr><th>Value</th><th></th><th>Metric</th><th>Value</th><th>IEC</th><th>JEDEC</th></tr><tr><td>1000</td><td>kB</td><td>kilobyte</td><td>1024</td><td>KiB</td><td>kibibyte</td></tr><tr><td>1000<sup>2</sup></td><td>MB</td><td>megabyte</td><td>1024<sup>2</sup></td><td>MiB</td><td>mebibyte</td></tr><tr><td>1000<sup>3</sup></td><td>GB</td><td>gigabyte</td><td>1024<sup>3</sup></td><td>GiB</td><td>gibibyte</td></tr><tr><td>1000<sup>4</sup></td><td>TB</td><td>terabyte</td><td>1024<sup>4</sup></td><td>TiB</td><td>tebibyte</td></tr><tr><td>1000<sup>5</sup></td><td>PB</td><td>petabyte</td><td>1024<sup>5</sup></td><td>PiB</td><td>pebibyte</td></tr><tr><td>1000<sup>6</sup></td><td>EB</td><td>exabyte</td><td>1024<sup>6</sup></td><td>EiB</td><td>exbibyte</td></tr></table> |                   |  | Multiples of bytes |  |  |  |  | Decimal |  |  | Binary |  | Value |  | Metric | Value | IEC | JEDEC | 1000 | kB | kilobyte | 1024 | KiB | kibibyte | 1000 <sup>2</sup> | MB | megabyte | 1024 <sup>2</sup> | MiB | mebibyte | 1000 <sup>3</sup> | GB | gigabyte | 1024 <sup>3</sup> | GiB | gibibyte | 1000 <sup>4</sup> | TB | terabyte | 1024 <sup>4</sup> | TiB | tebibyte | 1000 <sup>5</sup> | PB | petabyte | 1024 <sup>5</sup> | PiB | pebibyte | 1000 <sup>6</sup> | EB | exabyte | 1024 <sup>6</sup> | EiB | exbibyte |
| Multiples of bytes  |    |   |                   |  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| Decimal   |    |   | Binary            |  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| Value   |    | Metric  | Value             | IEC  | JEDEC              |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000  | kB | kilobyte  | 1024              | KiB  | kibibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000 <sup>2</sup>   | MB | megabyte  | 1024 <sup>2</sup> | MiB  | mebibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000 <sup>3</sup>   | GB | gigabyte  | 1024 <sup>3</sup> | GiB  | gibibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000 <sup>4</sup>   | TB | terabyte  | 1024 <sup>4</sup> | TiB  | tebibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000 <sup>5</sup>   | PB | petabyte  | 1024 <sup>5</sup> | PiB  | pebibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
| 1000 <sup>6</sup>   | EB | exabyte   | 1024 <sup>6</sup> | EiB  | exbibyte           |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |
|   |    | <b>Portability:</b> how easy the device can be carried<br><b>Durability:</b> how likely it is to survive in some extreme environment for example drops, water etc.<br><b>Reliability:</b> how long can the device keep working before inevitable failure  |                   |  |                    |  |  |  |  |         |  |  |        |  |       |  |        |       |     |       |      |    |          |      |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |          |                   |     |          |                   |    |         |                   |     |          |

|       |                           |                         |
|-------|---------------------------|-------------------------|
| Topic | 1.3 Hardware and Software | 1.3.6 Operating systems |
|-------|---------------------------|-------------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| describe the purpose of an operating system<br><i>(You will be required to understand the purpose and function of an operating system and why it is needed. You will not be required to understand how operating systems work.)</i> |
| show understanding of the need for interrupts   |

| Core Language   |  |   |
|---|--|---|
| <b>Operating Software:</b> operating software manages computer hardware, software resources, and provides common services for computer programs | <b>Scheduling:</b> the method that is used to distribute valuable computing resources, usually processor time, bandwidth and memory, to the various processes, threads, data flows and applications that need them | <b>Paging (memory management):</b> paging is a method used when creating virtual memory to use as RAM in the secondary memory, it controls data transmission between RAM and virtual memory |
| <b>Concurrent:</b> within this pattern, multiple calculations and processes are made within overlapping time                                    | <b>Authentication:</b> the process of verifying the identity of a person or device, for example logging in   | <b>User Interface:</b> the design for human-computer interaction and is the space where interactions  |

|   |   |   |
|---|---|---|
| frames  | with username and password  | between humans and machines occur   |
| <b>Interrupt:</b> an input signal to the processor indicating an event that needs immediate attention   | <b>Virtual Machine:</b> an emulation of a computer system based on computer architectures and provide functionality of a physical computer  | <b>WYSIWYG:</b> “what you see is what you get”, a user interface that allows the user to view something very similar to the end result—while the document is being created  |
| <b>GUI:</b> graphical user interface, allows users to interact with electronic devices through graphical icons<br>  | <b>CLI:</b> command-line interface, a text-based interface that is used to operate software and operating systems, processes commands to a computer program in the form of lines of text.<br>  | <b>System software:</b> System software is programs needed by the computer for it to run properly. They run in the background and normally users don't interact with them. These include operating systems, antivirus firewalls and defragmentation software.<br><b>Application software:</b> software directly used by users to carry out tasks such as word processing, browsing the internet and apps. |
| <b>Shareware:</b> shareware is software that is free to download and after a trial period usually of 30 days the user must pay to continue using it<br><br>Not free or editable(no source code)<br>Use for free for a period of time  | <b>Freeware:</b> This software can be downloaded for free but often only for personal use. The user cannot modify the software as they do not download the source code, but an executable file that's installed on the computer. An example of this is Google Chrome.<br><br>Free, shareable, not editable  | <b>Free software:</b> This comes with permission for the user to use, edit, copy or distribute the software for as long as they like but users cannot sell for their own financial gain. An example of this is Mozilla Firefox which is an open source browser.<br><br>Free, shareable, editable  |
| <b>Proprietary software:</b> This is developed by an organisation and sold to users with no access to the source code. Examples of these are Photoshop, Microsoft Office, Microsoft Windows. Occasionally the organisation will give away proprietary software for free such as Windows 10.<br><br>Not free, not editable | <b>Open source software/community:</b><br>Open source software is any software where the user is allowed to download the source code and then edit it to create different versions of the same software. Open source software often has very strong communities around it where people share knowledge and ideas. If you create a new version of an open source software you are expected to submit it back to the community for other people to use. An example of open source software is the operating system open Ubuntu. |   |

|  |  |  |
|--|--|--|
|  | It is often but not always free to download the source code. |  |
|--|--|--|

|       |                           |  |
|-------|---------------------------|--|
| Topic | 1.3 Hardware and Software | 1.3.7 High-level and low-level languages and their translators |
|-------|---------------------------|--|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| show understanding of the need for both high-level and low-level languages                              |
| show understanding of the need for compilers when translating programs written in a high-level language |
| show understanding of the use of interpreters with high-level language programs                         |
| show understanding of the need for assemblers when translating programs written in assembly language    |

| Core Language  |   |  |     |          |          |        |     |           |          |           |    |
|--|---|--|-----|----------|----------|--------|-----|-----------|----------|-----------|----|
| <p><b>High Level Language:</b> A high-level language is a computing language that humans use to code. It is highly abstract and requires a great deal of translation to become machine code.</p>   | <p><b>Low Level Language:</b> A low-level language is something like binary which is also called machine code. It has a very low level of abstraction and little translation needs to take place for the computer to be able to process it.</p>   | <p><b>Assembly Language:</b> assembly language is a low-level language written in mnemonics that closely reflects the operations of the CPU. Translated into machine code by assembler translator</p>  |     |          |          |        |     |           |          |           |    |
| <p><b>Translator:</b> source code must have been through translator that translate source code into machine code in order to be understood by the computer</p> <p><b>High &amp; low level language and machine code:</b></p> <div><div><div>High-Level Languages<br/>(Java, PHP, Python, etc.)</div><div>Assembly Language</div><div>Machine Code</div><div>Hardware</div></div><div><div>↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓</div><div>↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓</div></div><div><div>Assembler</div><div>Instruction set</div></div></div> | <p><b>Interpreter:</b> translates code into machine code line by line, the CPU executes each instruction before the interpreter moves on to translate the next instruction.</p> <p><b>Advantage &amp; Disadvantage:</b> Interpreted code will show an error as soon as it hits a problem, so it is easier to debug than compiled code. An interpreter does not create an independent final set of source code, the source code is created each time when it is run. Interpreted code is slower to execute than compiled code. And it always requires the interpreter of the coding language used in order to run the project.</p> <p><b>Examples:</b> Python, PHP</p> | <p><b>Compiler:</b> translates the whole program into machine code before the program is run and stored as a compiled object code separately</p> <p><b>Advantage &amp; Disadvantage:</b> It can be difficult to test individual lines of compiled code compared to interpreted languages as all bugs are reported after the program has been compiled. However, the machine code is saved and stored separately to the high-level code. Compilation is slow but compiled machine code can be executed quickly without interpreter or any translator.</p> <p><b>Examples:</b> Java, C++</p> |     |          |          |        |     |           |          |           |    |
| <p><b>Assembler:</b> translate assembly language into machine code</p>   | <p><b>Machine Code:</b> every types of code have to be translated into machine code so that the computer can understand ( binary )</p>  | <p><b>Source Code:</b> the original code of a program that is written in high level language</p>   |     |          |          |        |     |           |          |           |    |
| <p><b>Mnemonic:</b> assembly language instructions use abbreviations called mnemonics. Mnemonics are easier for humans to remember and understand than binary machine code instructions.</p>   |   |  |     |          |          |        |     |           |          |           |    |
| <table><tr><th>Assembly</th><th>Mnemonic</th><th>Binary</th><th>Hex</th></tr><tr><td>LOAD 0004</td><td>LDA 0004</td><td>1010 0100</td><td>A4</td></tr></table>   |   |  |     | Assembly | Mnemonic | Binary | Hex | LOAD 0004 | LDA 0004 | 1010 0100 | A4 |
| Assembly   | Mnemonic  | Binary   | Hex |          |          |        |     |           |          |           |    |
| LOAD 0004  | LDA 0004  | 1010 0100  | A4  |          |          |        |     |           |          |           |    |

|       |   |                         |
|-------|---|-------------------------|
| Topic | 1.2 Communication and Internet Technologies | 1.2.1 Data transmission |
|-------|---|-------------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| show understanding of what is meant by transmission of data   |
| distinguish between serial and parallel data transmission   |
| distinguish between simplex, duplex and half-duplex data transmission   |
| show understanding of the reasons for choosing serial or parallel data transmission   |
| show understanding of the need to check for errors  |
| explain how parity bits are used for error detection  |
| show understanding of the use of serial and parallel data transmission, in Universal Serial Bus (USB) and Integrated Circuit (IC) |

| Core Language   |  |  |                |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
|---|--|--|----------------|-------------------|-------------------------|--|------|-----|---------|---|----------|----------|---------|---|----------|----------|---------|---|----------|----------|---------|---|----------|----------|
| <b>Data:</b> 1s and 0s, saving information  | <b>Transmission:</b> the process of sending digital or analog data over a communication medium to another computer   | <b>Serial Transmission:</b> only one channel between the transmitter and the receiver<br>Easier to ensure that pieces of data are received in the correct order, cost less than parallel trans.  |                |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| <b>Parallel Transmission:</b> multiple channels between the transmitter and the receiver<br>faster than serial transmission   | <b>Simplex:</b> data transmission only available on one direction<br>For example: listening to broadcast   | <b>Half-Duplex:</b> bi-directional, non-simultaneous transmission<br>For example: walkie-talkie  |                |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| <b>Duplex:</b> bi-directional and simultaneous transmission<br>For example: social media  | <b>Error Detection:</b> the method of detecting errors that occur from the transmitter to the receiver   | <b>Parity Bit:</b> the extra bit added to the number unit when using parity check  |                |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| <b>Checksum:</b> use a specific algorithm to check whether the data remains the same after transmission<br>A work example<br>13784 would be transmitted as 137843 because the sum of each digits of this 5 digit number is 23, the last digit which is 3 is added to the end of the original number | <b>Majority Voting:</b><br>A method used in error detection that each bit is represented by three bits to fix damage from interference<br>A work example<br>010 111 01 would be transmitted as:<br>000 111 000 111 111 111 000 111 | <div>⬆ A work example of parity check:</div> <table><tr><th rowspan="2">7 bits of data</th><th rowspan="2">(count of 1-bits)</th><th colspan="2">8 bits including parity</th></tr><tr><th>even</th><th>odd</th></tr><tr><td>0000000</td><td>0</td><td>00000000</td><td>10000000</td></tr><tr><td>1010001</td><td>3</td><td>11010001</td><td>01010001</td></tr><tr><td>1101001</td><td>4</td><td>01101001</td><td>11101001</td></tr><tr><td>1111111</td><td>7</td><td>11111111</td><td>01111111</td></tr></table> | 7 bits of data | (count of 1-bits) | 8 bits including parity |  | even | odd | 0000000 | 0 | 00000000 | 10000000 | 1010001 | 3 | 11010001 | 01010001 | 1101001 | 4 | 01101001 | 11101001 | 1111111 | 7 | 11111111 | 01111111 |
| 7 bits of data  | (count of 1-bits)  | 8 bits including parity  |                |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
|   |  | even   | odd            |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| 0000000   | 0  | 00000000   | 10000000       |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| 1010001   | 3  | 11010001   | 01010001       |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| 1101001   | 4  | 01101001   | 11101001       |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |
| 1111111   | 7  | 11111111   | 01111111       |                   |                         |  |      |     |         |   |          |          |         |   |          |          |         |   |          |          |         |   |          |          |

|       |   |  |
|-------|---|--|
| Topic | 1.2 Communication and Internet Technologies | 1.2.3 Internet principles of operation |
|-------|---|--|

| Syllabus Learning Outcomes. You should be able to:  |
|---|
| show understanding of the role of the browser   |
| show understanding of the role of an Internet Service Provider (ISP)  |
| show understanding of what is meant by hypertext transfer protocol (http and https) and HTML                                  |
| distinguish between HTML structure(HTML) and presentation(CSS)  |
| show understanding of the concepts of MAC address, Internet Protocol (IP) address, Uniform Resource Locator (URL) and cookies |

| Core Language  |   |  |
|--|---|--|
| <b>Browser:</b> the piece of software that allows you to access web pages from the world wide web  | <b>HTML:</b> Hyper Text Markup Language, <b>the language of the website</b> , decides how the website is structured and how the website looks like  | <b>ISP:</b> internet service provider, companies provide service of accessing to the internet to their customers   |
| <b>Protocol:</b> rules that govern communication   | <b>HTTP:</b> Hyper Text Transfer Protocol, the protocol used for transmitting HTML for accessing and receiving web pages in form of HTTP files on the Internet  | <b>HTTPS:</b> HTTPS is an encrypted and secured version of HTTP  |
| <b>MAC Address:</b> a hardware identification number, it can uniquely identifies actual devices on a network and never change even if switched to another internet | <b>IP Address:</b> a set of numbers for communicating that identifies and locates the host and helps to transfer data over the internet   | <b>URL:</b> uniform resource locator, A URL is the unique web address for every webpage on the internet, instead of using long and complicated number-based IP address<br>E.g. "www.bbc.com" |
| <b>Cookie:</b> <b>a kind of spyware?</b> , small text files. They are stored on a user's computer to help websites remember preferences and track user's behaviour | <b>DNS:</b> domain name system, translate domain names into IP addresses, for example<br>bbc.com → 212.58.241.67<br>Difference between URL and domain name:<br>http://www.abzwebpedia.com/index.html is a complete URL. "abzwebpedia.com" is the domain name. | Protocols of accessing to the Internet:<br>Device -----MAC Address-----><br>Internet Hub -----//FIREWALL//-----<br>-----IP Address----->Internet   |



|       |   |                        |
|-------|---|------------------------|
| Topic | 1.2 Communication and Internet Technologies | 1.2.2 Security aspects |
|-------|---|------------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| show understanding of the security aspects of using the Internet and understand what methods are available to help minimise the risks |
| show understanding of the Internet risks associated with malware, including viruses, spyware and hacking                              |
| explain how anti-virus and other protection software helps to protect the user from security risks                                    |

| Core Language   |   |  |
|---|---|--|
| <b>Malware:</b> Malware, or malicious software is a program or file that is harmful to a computer, commonly including viruses, spywares and so on...  | <b>Phishing:</b> a technique used by hackers to acquire your personal information by sending an email that is designed to look just like a legitimate email and is intended to trick you into clicking on a malicious link or attachment. | <b>Virus(malware):</b> a simple program which is made to harm a computer system, it spreads by duplicating and attaching itself to files     |
| <b>Spyware:</b> a type of software that steal information (such as the usage of internet or other sensitive information) without the knowledge of the user  | <b>Hacking:</b> any unauthorised access to a computer system, the purpose can be good or bad  | <b>Cracking:</b> hacking a computer system and software for a malicious purpose, such as getting a password or people's personal information |
| <b>Pharming:</b> a form of online fraud involving malicious code and fraudulent websites. Cybercriminals install malicious code on your computer or server. The code automatically directs you to fake websites without your knowledge or consent.<br><a href="https://us.norton.com">Phishing vs pharming: https://us.norton.com</a> | <b>War Driving:</b> the act of searching for Wi-Fi wireless networks to do damage, usually in a moving vehicle  | <b>Shoulder Surfing:</b> to see someone's personal information (e.g. password PIN number) by physically looking over the person's shoulder   |
| <b>Black Hat Hacker:</b> using illegal ways for bad purposes (e.g. stealing information, damaging the system)   | <b>Grey Hat Hacker:</b> those who use illegal methods to hack into other people's system but with no malicious purpose<br>The condition between black and white hat hackers   | <b>White Hat Hacker:</b> often work with organizations to protect and ensure the security of the system with permission                      |
| <b>Firewall:</b> an application that prevents and cuts down unauthorised connections to and from the internet   | <b>Vulnerability Testing:</b> to detect and find the vulnerabilities within a system to check whether it is safe enough   | <b>Access Control:</b> the security system which makes decisions whether to approve or reject access requests via username, password etc.    |

|       |              |                           |
|-------|--------------|---------------------------|
| Topic | 1.4 Security | 1.4.1 Secure Data Storage |
|-------|--------------|---------------------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| show understanding of the need to keep data safe from accidental damage, including corruption and human errors                            |
| show understanding of the need to keep data safe from malicious actions, including unauthorised viewing, deleting, copying and corruption |

| Core Language   |   |  |
|---|---|--|
| <b>Encryption:</b> the process of changing electronic information or signals into a secret code                                 | <b>Durability:</b> how likely it is to withstand a damage from the outside such as water or drop        | <b>Reliability:</b> how long can the device keep working before inevitable failure |
| <b>Portability:</b> how easy is the data can be carried around; with cloud storage, portability starts to become not a big deal | <b>Data Corruption:</b> errors in computer data which introduce unintended changes to the original data | <b>Human Error:</b> errors made by people which cause accidental damage            |
| <b>HTTPS:</b> encrypted HTTP protocol to transfer HTML files  | <b>Accidental Damage:</b> unintended accidents which cause damage to the computer system                | <b>Hardware Faults:</b> failures occur itself to computer's hardwares              |
| <b>Software Faults:</b> failures occur while running a software   | <b>Spam:</b> unwanted emails or messages received   |  |

|       |              |  |
|-------|--------------|--|
| Topic | 1.4 Security | 1.4.2 Secure Data Transmission 1.4.3 Secure Online Systems |
|-------|--------------|--|

|  |
|--|
| <b>Syllabus Learning Outcomes. You should be able to:</b>  |
| <p>show understanding of how data are kept safe when stored and transmitted, including:</p> <ul style="list-style-type: none"> <li>• use of passwords, both entered at a keyboard and biometric</li> <li>• use of firewalls, both software and hardware, including proxy servers</li> <li>• use of security protocols such as Secure Socket Layer (SSL) and Transport Layer Security (TLS)</li> <li>• use of symmetric encryption (plain text, cypher text and use of a key) showing understanding that increasing the length of a key increases the strength of the encryption</li> </ul> |
| <p>show understanding of the need to keep online systems safe from attacks including denial of service attacks, phishing, pharming</p>   |
| <p>describe how the knowledge from this unit can be applied to real-life scenarios including, for example, online banking, shopping</p>  |

| Core Language   |   |  |
|---|---|--|
| <b>Strong password:</b> a combination of both uppercase and lowercase letters, numbers and other symbols  | <b>Hashing:</b> an algorithm that converts your password into fixed digits, so the server could store the password after hashing instead of your real password. Hashed numbers can hardly be decrypted.   | <b>Biometric:</b> examples are fingerprint, face recognition and eyeball recognition   |
| <b>Proxy Server:</b> a third party server that is used as a hub when you request to access the internet, it can also provide extra security   | <b>Hardware Firewall / Software Firewall:</b><br>Firewall is a network security feature that monitors and filters incoming and outgoing network traffic based on a white list or a black list. A hardware firewall protects you from the outside world, and a software firewall protects a specific device from other internal systems. |  |
| <b>SSL:</b> secure sockets layer, the older version of TLS  | <b>TLS:</b> transport layer security, it allows data to be sent and received securely over the internet by using authentication and encryption  | <b>Symmetric Encryption:</b> both the sender and the receiver agree to one public key that encrypts the data being transmitted, so that the original data are less likely to be seen by others during transmission |
| <b>Asymmetric Encryption:</b> every user has a private key while sharing a same public key with other users, the sender uses the public key to encrypt the data and the receiver uses both public key and their own private key to decrypt it | <b>DoS:</b> denial of service attack, an attack meant to shut down a network, making it inaccessible to its users. DoS attacks accomplish this by continuously sending useless access requests to the system and intending to overload it   | <b>DDoS:</b> distributed denial of service, multiple computers targeting one single system with a DoS attack   |

|       |            |            |
|-------|------------|------------|
| Topic | 1.5 Ethics | 1.5 Ethics |
|-------|------------|------------|

|   |
|---|
| <b>Syllabus Learning Outcomes. You should be able to:</b>   |
| show understanding of computer ethics, including copyright issues and plagiarism  |
| distinguish between free software, freeware and shareware   |
| show understanding of the ethical issues raised by the spread of electronic communication and computer systems, including hacking, cracking and production of malware |

| Core Language   |   |   |
|---|---|---|
| <b>Copyright:</b> the exclusive right received by the owner of some intellectual properties to make copies of their work.           | <b>Piracy:</b> the unauthorized use or reproduction of other's work   | <b>Plagiarism:</b> the practice of taking someone else's work or ideas and claim them as their own                            |
| <b>Freeware:</b> copyrighted software that is available at no cost for unlimited usage  | <b>Shareware:</b> software that is only offered for free for a certain period of time and after that the user has to pay for this software or service | <b>Open Source Software:</b> softwares with source code available for downloading so that anyone can inspect and make changes |
| <b>Proprietary Software:</b> the downloader can use it but cannot do any edit to the software itself, usually not provided for free | <b>Creative Commons License:</b> allowed the general public to copy or make changes to a file of the copyright holder<br>Example: wikipedia           |   |