|  |  |  |
| --- | --- | --- |
| **Paddington Academy Topic Overview**  **Subject: Computer Science**  **Exam Board: OCR H446** | | |
| **Topic** | **Sub-Topics** | **Textbook/Revision Guide Pages** |
| The characteristics of contemporary processors, input, output and storage devices | The Arithmetic and Logic Unit; ALU, Control Unit  and Registers (Program Counter; PC, Accumulator;  ACC, Memory Address Register; MAR, Memory Data  Register; MDR, Current Instruction Register; CIR).  Buses: data, address and control: how this relates to assembly language programs. |  |
| The fetch-decode-execute cycle, including its effect on registers. |  |
| The factors affecting the performance of the CPU, clock speed, number of cores, cache. |  |
| The use of pipelining in a processor to improve efficiency. |  |
| Von Neumann, Harvard and contemporary processor architecture. |  |
| The differences between and uses of CISC and RISC processors. |  |
| GPUs and their uses (including those not related to graphics). |  |
| Multicore and Parallel systems. |  |
| How different input, output and storage devices can be applied to the solution of different problems. |  |
| The uses of magnetic, flash and optical storage devices. |  |
| RAM and ROM. |  |
| Virtual storage. |  |
| Software and software development | The need for, function and purpose of operating systems. |  |
| Memory Management (paging, segmentation and virtual memory). |  |
| Interrupts, the role of interrupts and Interrupt Service Routines (ISR), role within the Fetch‑Decode-Execute Cycle. |  |
| Scheduling: round robin, first come first served,  multi-level feedback queues, shortest job first and shortest remaining time. |  |
| Distributed, embedded, multi-tasking, multi-user and real time operating systems. |  |
| BIOS |  |
| Device Drivers |  |
| Virtual machines, any instance where software is used to take on the function of a machine, including executing intermediate code or running an operating system within another |  |
| The nature of applications, justifying suitable applications for a specific purpose. |  |
| Open source vs closed source applications |  |
| Procedural programming language techniques: |  |
| Translators: Interpreters, compilers and assemblers. |  |
| Stages of compilation (lexical analysis, syntax analysis, code generation and optimisation). |  |
| Linkers and loaders and use of libraries. |  |
|  | Understand the waterfall lifecycle, agile methodologies, extreme programming, the spiral model and rapid application development. |  |
|  | The relative merits and drawbacks of different methodologies and when they might be used. |  |
|  | Writing and following algorithms. |  |
|  | Need for and characteristics of a variety of programming paradigms. |  |
|  | Procedural languages. |  |
|  | Assembly language (including following and writing simple programs with the Little Man Computer instruction set). See appendix 5e |  |
|  | Modes of addressing memory (immediate, direct, indirect and indexed). |  |
|  | Object-oriented languages (see appendix 5e for pseudocode style) with an understanding of classes, objects, methods, attributes, inheritance, encapsulation and polymorphism. |  |
| Exchanging data | Lossy v lossless compression. |  |
| Run length encoding and dictionary coding for lossless compression. |  |
| Symmetric and asymmetric encryption. |  |
| Different uses of hashing. |  |
| Relational database, flat file, primary key, foreign key, secondary key, entity relationship modelling. |  |
| Methods of capturing, selecting, managing and exchanging data. |  |
| Characteristics of networks and the importance of protocols and standards. |  |
| Normalisation to 3NF. |  |
| SQL – Interpret and modify. See appendix 5e. |  |
| Referential integrity. |  |
| Transaction processing, ACID (Atomicity, Consistency, Isolation, Durability), record locking and redundancy. |  |
| Characteristics of networks and the importance of protocols and standards. |  |
| Internet structure: The TCP/IP stack, DNS, Protocol layering, LANs and WANs, Packet and circuit switching. |  |
| Network security and threats, use of firewalls, proxies and encryption |  |
| Network Hardware |  |
| Client-server and peer to peer. |  |
| HTML, CSS and JavaScript. |  |
| Search Engine Indexing |  |
|  | PageRank algorithm |  |
|  | Server and Client side processing |  |
| Data types, data structures and algorithms | Primitive data types, integer, real/floating point, character, string and Boolean |  |
| Represent positive integers in binary. |  |
| Use of sign and magnitude and two’s complement to represent negative numbers in binary |  |
| Addition and subtraction of binary integers. |  |
| Represent positive integers in hexadecimal. |  |
| Convert positive integers between binary hexadecimal and denary. |  |
| Representation and normalisation of floating point numbers in binary. |  |
| Floating point arithmetic, positive and negative numbers, addition and subtraction. |  |
| Bitwise manipulation and masks: shifts, combining with AND, OR, and XOR. |  |
| How character sets (ASCII and UNICODE) are used to represent text. |  |
| Arrays (of up to 3 dimensions), records, lists, tuples |  |
| The following structures to store data: linked-list, graph (directed and undirected), stack, queue, tree, binary search tree, hash table. |  |
| How to create, traverse, add data to and remove data from the data structures mentioned above. (NB this can be either using arrays and procedural programming or an object-oriented approach). |  |
| Define problems using Boolean logic. |  |
| Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions. |  |
| Use the following rules to derive or simplify statements in Boolean algebra: De Morgan’s Laws, distribution, association, commutation, double negation. |  |
| Use logic gate diagrams and truth tables. |  |
|  | The logic associated with D type flip flops, half and full adders |  |
| Legal, moral, ethical and cultural issues | The Data Protection Act 1998. |  |
| The Computer Misuse Act 1990. |  |
| The Copyright Design and Patents Act 1988. |  |
| The Regulation of Investigatory Powers Act 2000. |  |
| The individual moral, social, ethical and cultural opportunities and risks of digital technology |  |
| Computers in the workforce |  |
| Automated decision making. |  |
| Artificial intelligence. |  |
| Environmental effects. |  |
| Censorship and the Internet. |  |
| Elements of computational thinking | The nature of abstraction. |  |
| The need for abstraction. |  |
| The differences between an abstraction and reality. |  |
| The differences between an abstraction and reality. |  |
| Identify the inputs and outputs for a given situation |  |
| Determine the preconditions for devising a solution to a problem. |  |
| The need for reusable program components. |  |
| Identify the components of a problem |  |
| Determine the order of the steps needed |  |
| Identify the points in a solution where a decision has to be taken |  |
| Determine the logical conditions that affect the outcome of a decision. |  |
|  | Determine how decisions affect flow through a program. |  |
|  | Determine the parts of a problem that can be tackled at the same time. |  |
|  | Outline the benefits and trade offs that might result from concurrent processing in a particular situation. |  |
| Problem solving and programming | Programming constructs: sequence, iteration, branching |  |
| Recursion, how it can be used and compares to an iterative approach |  |
| Global and local variables. |  |
| Modularity, functions and procedures, parameter passing by value and reference. |  |
| Use of an IDE to develop/debug a program. |  |
| Use of object oriented techniques. |  |
| Features that make a problem solvable by computational methods. |  |
| Problem recognition. |  |
| Problem decomposition |  |
| Use of divide and conquer. |  |
| Use of abstraction. |  |
| Learners should apply their knowledge of: backtracking/data mining/heuristics/performance modelling/pipelining/visualisation to solve problems. |  |
| Algorithms | Analysis and design of algorithms for a given situation. |  |
| The suitability of different algorithms for a given task and data set, in terms of execution time and space. |  |
| Measures and methods to determine the efficiency of different algorithms, Big O notation (constant, linear, polynomial, exponential and logarithmic complexity). |  |
| Comparison of the complexity of algorithms. |  |
| Algorithms for the main data structures, (stacks, queues, trees, linked lists, depth-first (post-order) and breadth-first traversal of trees). |  |
| Standard algorithms (bubble sort, insertion sort, merge sort, quick sort, Dijkstra’s shortest path algorithm, A\* algorithm, binary search and linear search). |  |