

## Definitions and Concepts for AQA Computer Science AS-level

### Topic 6: Fundamentals of Computer Systems

---

#### 6.1 Hardware and Software

##### 6.1.1 Relationship Between Hardware and Software

**Hardware:** The physical components of a computer system, including both external (peripheral) and internal (processing and storage) parts.

**Software:** Any program or collection of instructions and data that can be run and processed by a computer system.

##### 6.1.2 Classification of Software

**Application Software:** A program that can be run on a computer, allowing the user to carry out specific tasks.

**System Software:** A program designed to cover technical aspects of setting up, running and maintaining a computer system, and providing a platform for application software.

##### 6.1.3 System Software

**Assemblers:** A translator in low level language, which converts assembly language into machine code.

**Compilers:** A translator that converts high level language to machine code.

**Interpreters:** A translator which checks a source program for syntax errors line by line, translates it to machine code and executes the line.

**Libraries:** A collection of programs which are already compiled and can be loaded into a program and run whenever required.

**Operating System:** A set of programs managing the operation of the computer that is loaded into RAM everytime the computer is turned on. It bridges the user to the hardware.

**Translator:** A program which converts code from one computer language to another.

**Utility Program:** A program made to perform a generic or common task that is routinely executed by a user, related to analysing, configuring or optimizing.

This work by [PMT Education](https://www.pmt.education) is licensed under [CC BY-NC-ND 4.0](https://creativecommons.org/licenses/by-nc-nd/4.0/)



### 6.1.4 Role of an Operating System (OS)

**Resource Management:** The collective efficient management of the available hardware and software to optimise the performance of the computer system.

**Scheduling:** Allocating processor time to each application to ensure processor time is used as efficiently as possible when multitasking.

## 6.2 Classification of Programming Languages

**Assembly Language:** A low-level programming language consisting of a set of mnemonic instructions that directly corresponds to the processor architecture's machine code instruction set.

**High-Level Language:** A programming language with a strong abstraction from a processor's internal instruction set that is much more human-readable with natural-language keywords, such as Python or Java.

**Imperative Languages:** A programming language built on the programming paradigm of using subroutines and procedures as instructions to change a program's state and describe how a program operates.

**Low-Level Language:** A programming language with little to no abstraction from a processor's internal instruction set, such as machine code or assembly language.

**Machine Code:** A low-level programming language written in binary that is directly understood by the CPU.

## 6.3 Types of Program Translators

**Bytecode:** An intermediate instruction set used to write the final output of some compilers, since it can be executed on any computer via a virtual machine.

## 6.4 Logic Gates

**AND ( $\wedge$ ):** A logical operator which returns TRUE (or 1) if and only if all inputs are TRUE (or 1).

**OR ( $\vee$ ):** A logical operator which returns TRUE (or 1) if and only if at least one of the inputs are TRUE (or 1).

**XOR:** A logical operator which returns TRUE (or 1) if and only if exactly 1 of the inputs are TRUE (or 1).

**NOT ( $\neg$ ):** A logical operator which returns TRUE (or 1) if and only if the input is FALSE (or 0), i.e. it returns the opposite of the input.



**NAND:** A logical operator which returns FALSE (or 0) if and only if all inputs are TRUE (or 1). It is equivalent to an AND gate connected to a NOT gate.

**NOR:** A logical operator which returns FALSE (or 0) if and only if at least one of the inputs are TRUE (or 1). It is equivalent to an OR gate connected to a NOT gate.

## **6.5 Boolean Algebra**

**Absorption Laws:**  $A \wedge (A \vee B) = A$ ,

$$A \vee (A \wedge B) = A$$

**Association Laws:**  $A \wedge (B \wedge C) = (A \wedge B) \wedge C$ ,

$$A \vee (B \vee C) = (A \vee B) \vee C$$

**Boolean Expressions:** A combination of boolean variables and logical operators which evaluates to either TRUE or FALSE depending on the input.

**Boolean Logic:** A type of algebra with logical operators where all values and expressions ultimately reduce to TRUE or FALSE.

**Commutation Laws:**  $A \wedge B = B \wedge A$ ,

$$A \vee B = B \vee A$$

**De Morgan's First Law:**  $\neg(A \vee B) = \neg A \wedge \neg B$

**De Morgan's Second Law:**  $\neg(A \wedge B) = \neg A \vee \neg B$

**Distribution Laws:**  $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$ ,

$$(A \vee B) \wedge (C \vee D) = (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D)$$

**Double Negation Law:**  $A = \neg\neg A$

**Idempotence Laws:**  $A \wedge A = A$ ,

$$A \vee A = A$$

**Inverse Laws:**  $A \wedge \neg A = 0$ ,

$$A \vee \neg A = 1$$

