AQA Computer Science A-Level
4.6.2 Classification of programming languages
Concise Notes

Specification:

4.6.2.1 Classification of programming languages:

Show awareness of the development of types of programming languages and their classification into low-and high-level languages.

Know that low-level languages are considered to be:

- machine-code
- assembly language

Know that high-level languages include imperative high-level language.

Describe machine-code language and assembly language.

Understand the advantages and disadvantages of machine-code and assembly language programming compared with high-level language programming.

Explain the term ‘imperative high-level language’ and its relationship to low-level languages.

The development of types of programming languages

- Early computers could only be programmed using low-level languages.
- These languages:
  - directly manipulated the processor
  - required a great deal of effort from the programmer
  - were prone to errors
- High-level languages were developed, making the job of programming easier

Low-level languages

- Programs are processor specific
- Programs directly affect the computer's processor
- There are two categories:
  - machine code
  - assembly language

Machine code
- Uses only the binary digits 1 and 0 to represent instructions
- Directly manipulates a computer's processor
- Programs are very long and extremely difficult for humans to understand
- Programs are therefore prone to errors and difficult to debug
- Useful for embedded systems and real-time applications

Assembly language
- Developed to simplify the process of writing computer programs
- Mnemonics are used in place of the binary instructions that machine code uses
- Assembly language is therefore more compact and less error prone than machine code
- Each instruction has a 1-to-1 correlation to a machine code instruction
- AQA have made their own assembly language for use in exams

Machine code	Assembly language
`01010101`	`STR R4, #45`
`11010110`	`ADD R1, R2, 3`
`01001011`	`MOV R2, R1`
`10110110`	`HALT`

High-level languages

- Not platform specific
- Must be translated into machine code by a compiler or an interpreter before execution
- Use English instructions and mathematical symbols in instructions
- Examples include C#, Java, Pascal, Python and VB.Net
- Easy to learn, understand and debug
- Have built-in functions which can save vast amounts of time when programming
- Features like named variables, indentation and commenting make programs written in high-level languages easy to debug

Imperative high-level languages

- Formed from instructions that specify how the computer should complete a task, rather than what a computer should do

High-level languages vs. low-level languages

	Low-level		High-level
	Machine code	Assembly language	
Portability	Not portable. Programs are processor specific.		Portable. Programs are not specific to certain processors.
Ease of use	Code is difficult for humans to understand.	Mnemonics help to make code slightly easier for humans to understand.	Code uses English, making it easy for humans to understand.
Ease of debugging	Errors are very difficult to spot and correct.	Debugging is easier than with machine code but still far more difficult than with high-level languages.	Named variables, indentation and commenting make debugging fairly easy.
Ease of execution	Machine code is directly executed by processors.	An assembler must be used before a program is executed, but each instruction has a 1-to-1 correlation to a machine code instruction so translation is quick.	A compiler or interpreter must be used to translate source code into object code before it can be executed. This can be time consuming.