AQA Computer Science A-Level

4.6.3 Types of program translator

Concise Notes

Specification:

4.6.2.1 Classification of programming languages:

Understand the role of each of the following:

- Assembler
- Compiler
- Interpreter

Explain the differences between compilation and interpretation.

Describe situations in which each would be appropriate.

Explain why an intermediate language such as bytecode is produced as the final output by some compilers and how it is subsequently used.

Understand the difference between source code and object (executable) code.

Types of program translator

- There are three types of program translator:
  - Assemblers
  - Compilers
  - Interpreters

Assemblers

- Translate assembly language into machine code
- Are platform specific
- Each assembly language instruction has a 1-to-1 relationship to a machine code instruction
- Translation is fairly quick and straightforward

Compilers

- Translate high-level languages into machine code
- Are platform specific
- Take a high-level program as their source code
- Check the source code for any errors line-by-line
- Translate the entire program at once
- If the source code contains an error, it will not be translated.
- Compiled programs can be run without any other software present

Interpreters

- Translate high-level languages into machine code
- Translate line-by-line
- Have procedures to translate each kind of program instruction
- Check for errors as they translate
- Can be partially translate source code containing errors
- Both the program source code and the interpreter itself must be present
- This results in poor protection of the source code

Comparison of compilers and interpreters

Compiler	Interpreter
Checks source code for errors line-by-line before translating	Translation begins immediately
Entire source code translated at once	Each line is checked for errors and then translated sequentially
No need for source code or compiler to be present when the translated code is executed	Both the source code and the interpreter must be present when the program is executed
Protects the source code from extraction	Offers little protection of source code

Compilers with intermediate languages

- Don't produce machine code straight away
- Translate source code into an intermediate language, often bytecode
- A virtual machine is used to execute the bytecode on different processors
- Each different processor instruction set has its own virtual machine
- Allows for platform independence

Source code and object code

- Source code = input to a translator
- Object code = translator's output