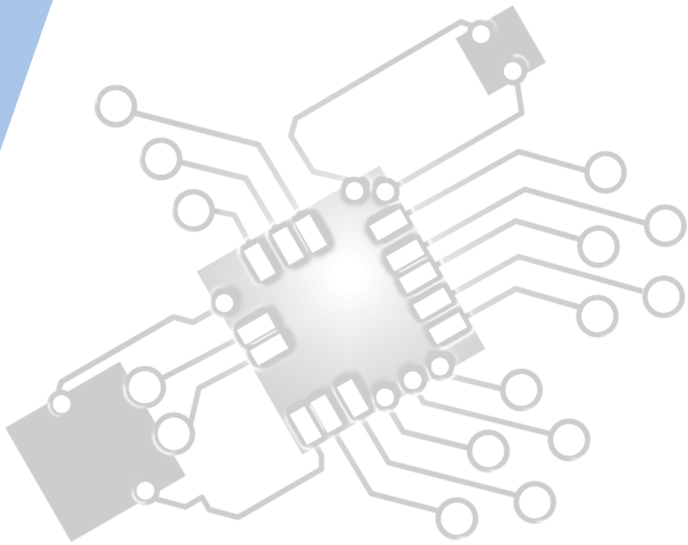
Computational thinking, problem-solving and programming:

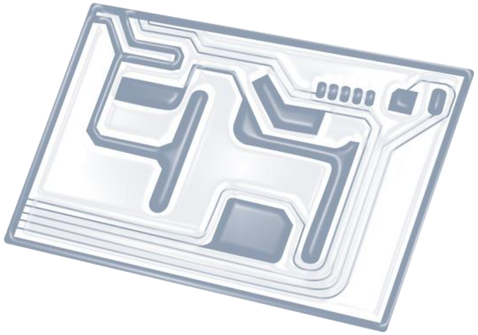
Introduction to programming

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

Content developed by Dartford Grammar School Computer Science Department

HL & SL 4.3 Overview

Nature of programming languages

4.3.1 State the fundamental operations of a computer

4.3.2 Distinguish between fundamental and compound operations of a computer

4.3.3 Explain the essential features of a computer language

4.3.4 Explain the need for higher level languages

4.3.5 Outline the need for a translation process from a higher level language to machine executable code

Use of programming languages

4.3.6 Define the terms: variable, constant, operator, object

4.3.7 Define the operators =, ., <, <=, >, >=, mod, div

4.3.8 Analyse the use of variables, constants and operators in algorithms

4.3.9 Construct algorithms using loops, branching

4.3.10 Describe the characteristics and applications of a collection

4.3.11 Construct algorithms using the access methods of a collection

4.3.12 Discuss the need for sub-programmes and collections within programmed solutions

4.3.13 Construct algorithms using predefined sub-programmes, one-dimensional arrays and/or collections

1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

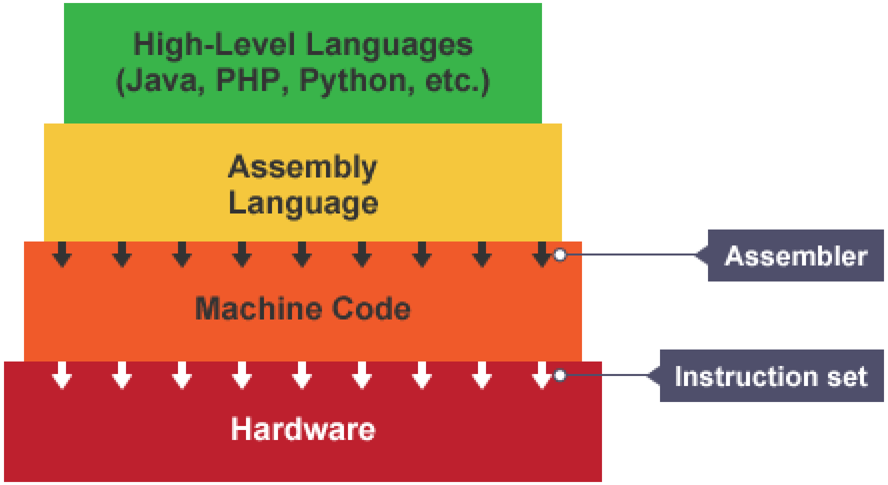
7: Control

D: OOP

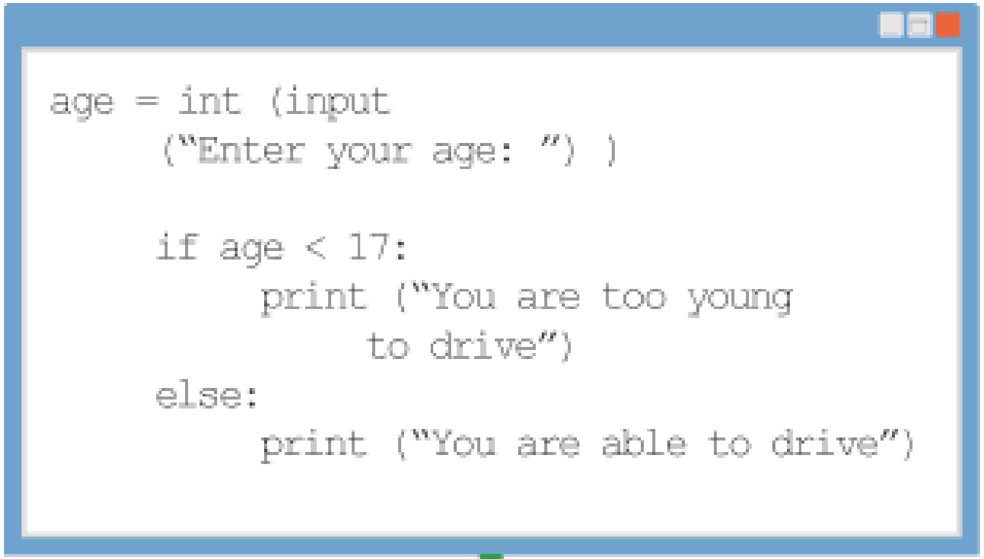
Content developed by Dartford Grammar School Computer Science Department

Topic 4.3.5

Outline the need for a **translation** process from a higher level language to machine executable code



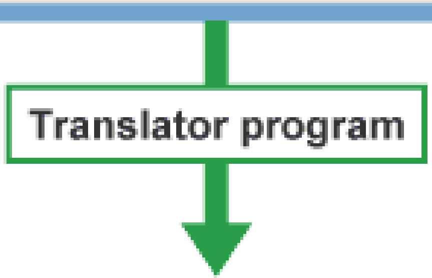
Content developed by Dartford Grammar School Computer Science Department



```
age = int (input
     (“Enter your age: ”) )

     if age < 17:
          print (“You are too young
               to drive”)
     else:
          print (“You are able to drive”)
```

Program is written in high level language



Translator program

```
0   1   0   1       1   1   0   1

1   0   1   0       0   1   1   0

0   1   0   1       1   0   0   1

1   1   1   0       0   1   0   0
```

Machine code is produced

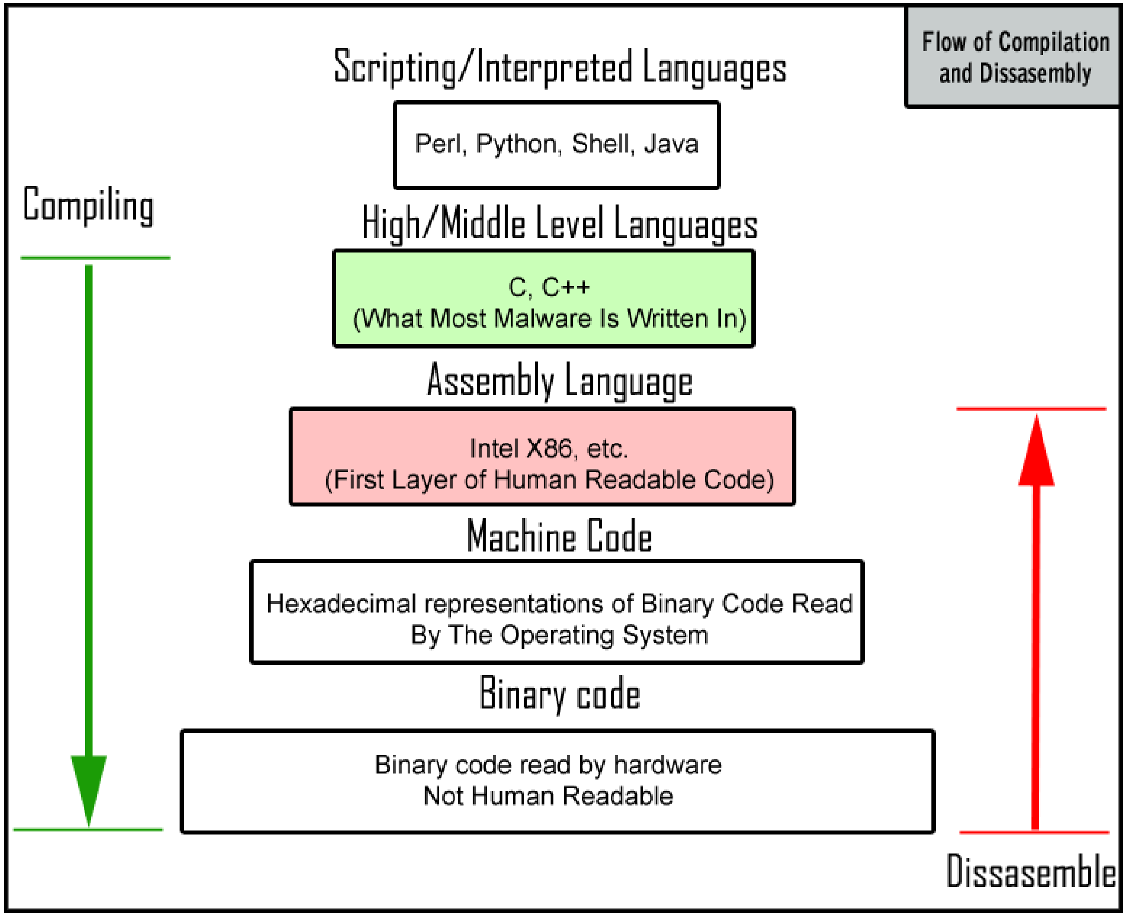
Video: Interpreter vs Compiler

https://www.youtube.com/watch?v=_C5AHaS1mOA#t=44

Key terms

- **Compiler** - If the translator translates a high level language into a lower level language (*done in a batch*)
- **Interpreter** - the translator translates a high level language into an intermediate code which will be immediately executed by the CPU (*done line by line*)
- **Assembler** - the translator translates assembly language to machine code (*mnemonics to binary*)

Levels of language



Flow of Compilation and Dissasembly

Scripting/Interpreted Languages

Perl, Python, Shell, Java

High/Middle Level Languages

C, C++
(What Most Malware Is Written In)

Assembly Language

Intel X86, etc.
(First Layer of Human Readable Code)

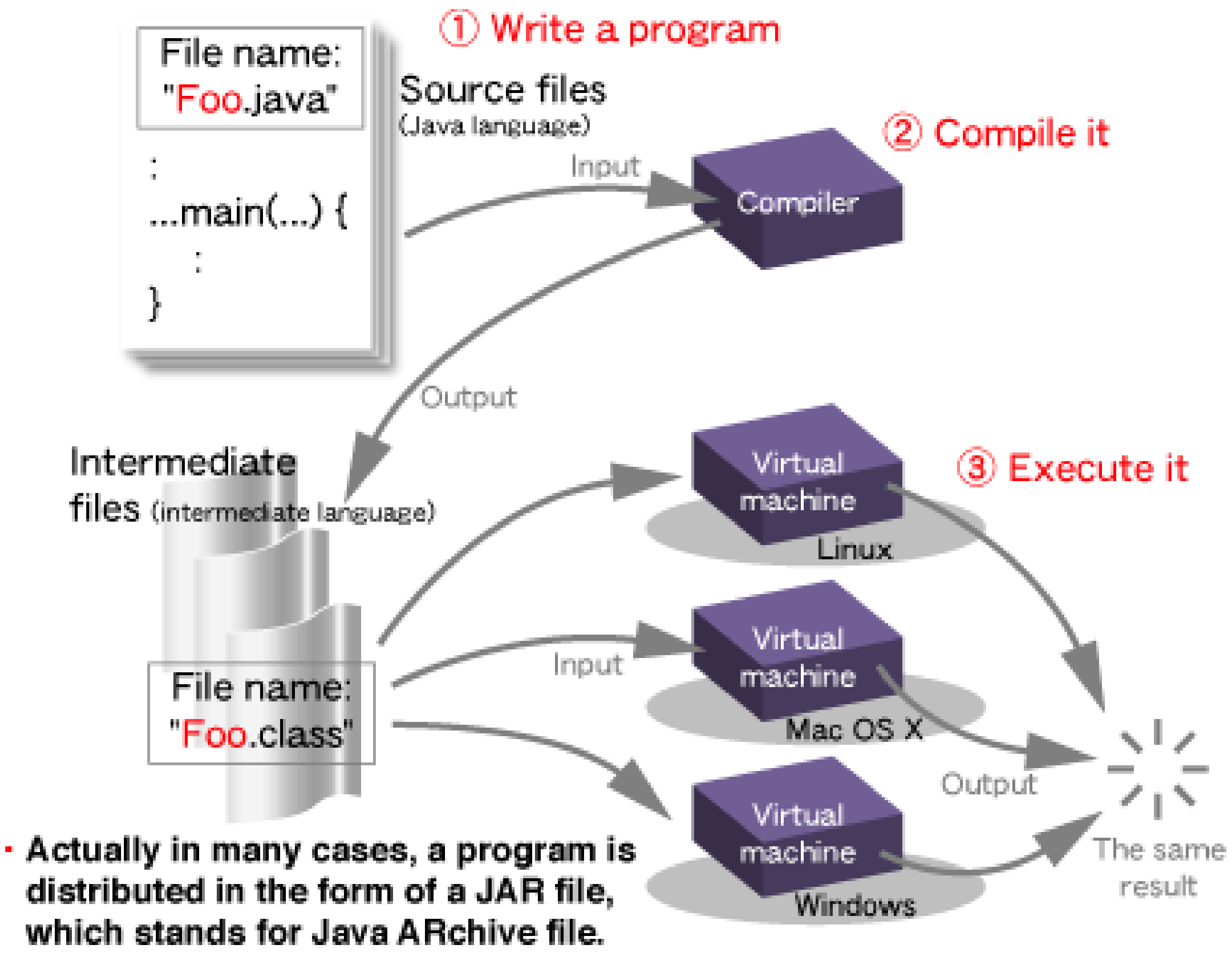
Machine Code

Hexadecimal representations of Binary Code Read By The Operating System

Binary code

Binary code read by hardware
Not Human Readable

The Java ‘process’: VM!



- Actually in many cases, a program is distributed in the form of a JAR file, which stands for Java ARchive file.

Java virtual machine

- ✓ **Java applications run on a virtual machine** (the Java Virtual Machine or JVM).
- ✓ This virtual machine is installed on the computer (e.g. PC, Mac, Smart Phone, Ticket Machine) and allows the **same java co**de to be run on **many different types of hardware**.
- ✓ Even though the hardware **architecture** and **instruction sets** of each of these devices is different the virtual machine is the same.
- ✓ The trick is that the virtual machine software needs to **match the hardware** it will be installed on, so you need to get the correct version of the virtual machine, but once you have it then you can run **any java program**.
- ✓ This is **good for the java programmer** as he does not have to write lots of different versions of the program he is writing, for each of the devices he wants it to run on.
- ✓ After the java program is written it can be deployed to **any device** that has the Java Virtual Machine installed on it.