Programming languages

High level programming languages

- Closer to human language and is therefore easier to understand.
- A translator is used to convert the instructions into code that the computer understand
- Allow programs to be written that is independent of the type of computer system
- It is up to the compiler to translate the code into the right machine code for a particular code.

Low level programming languages

Machine code is written in binary (1) so instructions can be processed **directly** by the CPU / do not need to be translated

Assembly language.
Assembly language requires an assembler to translate the code to binary

- It is difficult for humans to understand.
- Appropriate for developing new operating systems, embedded systems and hardware device drivers
- They are microprocessor/CPU/machine specific (1) so they can manipulate the hardware directly (1)
- They can be highly optimised (1) to make efficient use of the hardware/execute more quickly/use minimal memory (1)
- Each line of code (1) is one instruction only (1)

Assembly Language

Mnemonics are used to represent an instructions

Assembly language sample Instruction set
LOAD #23 # Load from RAM to processor
MOV a 23 # Transfer in number 23 into the variable a
ADD 2 3 # Add 2 values
STORE # store data in RAM

Machine code

- is expressed in binary values 0 and 1.
- This is the language that computers understand.
- All codes whether assembler or high level programming languages need to be translated into machine code.
- Machine code is specific to a processor.

Machine code instructions are made up of two parts:

- **Operator – instruction (eg. Add, load).**
- **Operand - value or memory address**

Machine code instruction	
Operator	Operand
0011	10010100

Model Answer

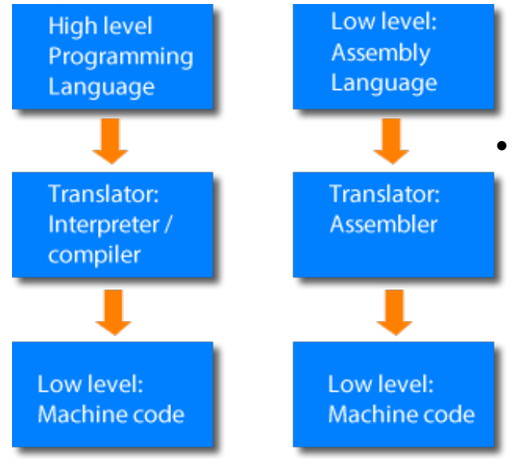
What is the difference between high and low level languages?

- High-level languages use instructions that look like English (1), whereas a low-level language uses mnemonics/binary code (1).
- A high-level language statement generates many lines of machine code
- (1), whereas each line of low-level languages is/generates a single machine instruction (1)
- High-level languages are general purpose/exist across microprocessors/CPUs/machine-independent (1), whereas a low-level
- language is microprocessor/CPU/machine specific (1)
- High-level languages are abstracted from the hardware (1), whereas
- low-level languages manipulate the hardware directly (1)

Program translators

Program translators allow programs to be translated into machine code so the than programs can be run on a computer.

Interpreter



- converts high level languages into machine code one instruction at a time on-the-fly while the program is running.
- Each instruction is converted to machine code once the previous instruction has been executed.
- Interpreters are good for debugging code because the program stops as soon as the error has been found.
- Much **slower** running compiled code.

Compiler

- A program that converts high level languages into machine code before the program is run.
- A compiler saves the machine code, so the source code is no longer needed
- A compiler allows a program to be **run faster** than interpreted code.
- Software is normally distributed as compiled machine code. For proprietary software this is good because other people cannot copy
- the code and use it for their own applications.

Model Answer:

- *Translates the entire source file to machine code in one go/translates all the source code prior to execution (1)*
- *Shows all syntax errors at the end of the translation process (1)*
- *Produces a single executable/object code/platform-dependent output (1)*
- *Separates the tasks of translation and execution (1)*
- Assembler converts assembly language instructions into machine code.
- *A compiler carries out translation once prior to execution (1)*
- *whereas an interpreter carries out translation every time the*
- *program executes (1)*
- *A compiler produces a stand-alone executable file (1) whereas an interpreter is required each time the code is run (1)*
- *A compiler reports errors after translation is complete (1) whereas an interpreter reports errors as they occur (1)*