ANSWERS

F6.5 Languages and translators

Robust programs · GCSE · OCR J277 2.5.1, AQA 8525 3.4.4, Edexcel 1CP2 3.3.1 · about 20 min

What this lesson is about

High and low level, machine code and assembly; compilers, interpreters and assemblers.

- High level and low level
- Assembly language
- Translators
- Which does BugBot use?
- An interpreter of your own

Questions

6 marks in all

1. Why must a Python program be translated before it runs? [1 mark]

- ☐ **A** A processor only understands its own machine code
- ☐ **B** Python is too slow
- ☐ **C** To remove comments
- ☐ **D** To check spelling

> **Answer:** A. High-level code has to become machine code, either all at once or a statement at a time.

2. Which translator turns assembly language into machine code? [1 mark]

- ☐ **A** An assembler
- ☐ **B** A compiler
- ☐ **C** An interpreter
- ☐ **D** An IDE

> **Answer:** A. An assembler translates each assembly instruction into one machine code instruction.

3. Which translator produces a standalone file that runs without it? [1 mark]

- ☐ **A** A compiler
- ☐ **B** An interpreter
- ☐ **C** An editor
- ☐ **D** A debugger

> **Answer:** A. A compiler translates the whole program in one go into an executable file.

4. Why is an interpreter useful while developing a program? [1 mark]

- ☐ **A** It stops at the first error, on the line where it happened
- ☐ **B** It makes the finished program run fastest
- ☐ **C** It produces an executable file
- ☐ **D** It hides the source code

> **Answer:** A. Translating and running a line at a time makes errors easy to find and fix.

5. Which are features of low-level languages? [1 mark]

Tick every answer that is true.

- ☐ **A** One instruction for each processor instruction
- ☐ **B** Written for a particular kind of processor
- ☐ **C** Close control of memory and hardware
- ☐ **D** Easy to read and move between computers

> **Answer:** A, B, C. Portability and readability are advantages of high-level languages.

6. The robot assembly program is LOAD 4, FWD 20, TURN 90, BEEP 660, DEC, JNZ 1, HALT. How many beeps does it play? [1 mark]

> **Answer:** 4. R starts at 4; each time round DEC takes one off, and JNZ jumps back until R is 0.

The task: robot assembly

Finish the interpreter so it runs the robot assembly program below, which uses `LOAD`, `DEC` and `JNZ` to drive a square with a beep at each corner. Store the value of `R` in a variable, and make `JNZ` change the program counter.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

program = ["LOAD 4", "FWD 20", "TURN 90", "BEEP 660", "DEC", "JNZ 1", "HALT"]

pc = 0
while True:
    op, *args = program[pc].split()
    if op == "FWD":
        forward(60, distance=int(args[0]))
    elif op == "TURN":
        turn_right(30, angle=int(args[0]))
    elif op == "HALT":
        break
    pc = pc + 1
```

The hint students can ask for: Keep the register in a variable. Each instruction reads the word and acts on it. The jump instruction is the only one that changes where the program counter goes next, so make sure it does not then get moved on again.

A solution

```
from bugbot import *
connect()
program = ["LOAD 4", "FWD 20", "TURN 90", "BEEP 660", "DEC", "JNZ 1", "HALT"]

pc = 0
r = 0
while True:
    op, *args = program[pc].split()
    if op == "LOAD":
        r = int(args[0])
    elif op == "FWD":
        forward(60, distance=int(args[0]))
    elif op == "TURN":
        turn_right(30, angle=int(args[0]))
    elif op == "BEEP":
        tone(int(args[0]), 0.2)
    elif op == "DEC":
        r = r - 1
    elif op == "JNZ":
        if r != 0:
            pc = int(args[0])
            continue
    elif op == "HALT":
        break
    pc = pc + 1
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.