ANSWERS

F9.3 Hardware, software and von Neumann

Logic and computer systems · GCSE · OCR J277 1.1.1, AQA 8525 3.4.1, Edexcel 1CP2 3.1.1 · about 15 min

What this lesson is about

Hardware and software, and the stored program concept in a tiny computer.

- Hardware and software
- The stored program concept
- Program and data in one memory
- Why it matters

Questions

5 marks in all

1. Which of these is software? [1 mark]

- ☐ **A** The robot's firmware
- ☐ **B** The camera
- ☐ **C** The battery
- ☐ **D** The processor

> **Answer:** A. Software is the programs; the others are physical parts.

2. What is the stored program concept? [1 mark]

- ☐ **A** Program instructions and data are held together in the same memory
- ☐ **B** Programs are stored on a separate disk from data
- ☐ **C** Programs are wired into the processor
- ☐ **D** Every program is saved before it runs

> **Answer:** A. Loading a new program into memory changes the job, with no rewiring.

3. Which is system software? [1 mark]

- ☐ **A** An operating system
- ☐ **B** A web browser
- ☐ **C** A game
- ☐ **D** A word processor

> **Answer:** A. System software runs the computer; applications do jobs for the user.

4. What does this program print? [1 mark]

```
memory = ["LOAD 3", "ADD 4", "HALT", 30, 12]
acc = memory[int(memory[0].split()[1])]
acc = acc + memory[int(memory[1].split()[1])]
print(acc)
```

> **Answer:**
>
> 42
>
> LOAD 3 gets 30, ADD 4 adds 12.

5. How does the processor know which address holds the next instruction? [1 mark]

- ☐ **A** The program counter holds its address
- ☐ **B** Instructions are marked differently from data
- ☐ **C** It reads every address in order until it finds one
- ☐ **D** The user types it in

> **Answer:** A. Memory just holds values; the program counter points to the next instruction.

The task: a stored program

Add a `SUB` instruction to the tiny computer, which subtracts the value at an address from the accumulator. Then put a program in memory that loads 50 from address 6, subtracts 8 from address 7, stores the answer at address 8 and outputs it, so the program prints `output: 42`. The robot should then drive forward the answer in centimetres, divided by 2, read from memory address 8.

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

memory = ["LOAD 6", "ADD 7", "STORE 8", "OUT 8", "HALT", "", 50, 8, 0]
```

The hint students can ask for: Add one more branch to the processor loop for subtracting, and change the program in memory so it subtracts instead of adds. After the loop ends, read the answer back out of memory and use it to work out how far to drive.

A solution

```
from bugbot import *
connect()
memory = ["LOAD 6", "SUB 7", "STORE 8", "OUT 8", "HALT", "", 50, 8, 0]
acc = 0
pc = 0
while True:
    op, *arg = memory[pc].split()
    pc = pc + 1
    if op == "LOAD":
        acc = memory[int(arg[0])]
    elif op == "ADD":
        acc = acc + memory[int(arg[0])]
    elif op == "SUB":
        acc = acc - memory[int(arg[0])]
    elif op == "STORE":
        memory[int(arg[0])] = acc
    elif op == "OUT":
        print("output:", memory[int(arg[0])])
    elif op == "HALT":
        break
forward(50, distance=memory[8] / 2)
```

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