ANSWERS

F9.4 The CPU and fetch-execute

Logic and computer systems · GCSE · OCR J277 1.1.1, AQA 8525 3.4.5, Edexcel 1CP2 3.1.1 · about 20 min

What this lesson is about

The ALU, control unit, registers and buses, and tracing the fetch-decode-execute cycle.

- Inside the CPU
- Buses
- The fetch-decode-execute cycle
- Instructions are numbers too

Questions

6 marks in all

1. Which register holds the address of the next instruction to fetch? [1 mark]

- ☐ **A** Program counter
- ☐ **B** Memory data register
- ☐ **C** Accumulator
- ☐ **D** Memory address register

> **Answer:** A. The PC points at the next instruction.

2. What does the ALU do? [1 mark]

- ☐ **A** Carries out arithmetic and logic operations
- ☐ **B** Decodes instructions
- ☐ **C** Holds the address to read from
- ☐ **D** Sends the clock pulse

> **Answer:** A. Arithmetic logic unit: adding, subtracting, comparing.

3. Put these steps of the cycle in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `The instruction is executed`
___ `The address in the PC is copied to the MAR`
___ `The instruction at that address is copied to the MDR`
___ `The control unit decodes the instruction`

> **Answer:**
>
> ```
> The address in the PC is copied to the MAR
> The instruction at that address is copied to the MDR
> The control unit decodes the instruction
> The instruction is executed
> ```
>
> Fetch, decode, execute, then back to fetch.

4. Which bus carries addresses from the CPU to memory? [1 mark]

- ☐ **A** Address bus
- ☐ **B** Data bus
- ☐ **C** Control bus
- ☐ **D** USB

> **Answer:** A. The address bus goes one way: CPU to memory.

5. What does the memory data register hold? [1 mark]

- ☐ **A** The data or instruction just read from memory, or about to be written
- ☐ **B** The address of the next instruction
- ☐ **C** The result of the last calculation only
- ☐ **D** The clock speed

> **Answer:** A. The MDR is the CPU's end of the data bus.

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

```
pc = 4
mar = pc
pc = pc + 1
print(mar, pc)
```

> **Answer:**
>
> ```
> 4 5
> ```
>
> The MAR gets the old PC value, then the PC moves on.

The task: trace the cycle

Complete the fetch-decode-execute loop so it runs the program in `memory` and supports `LOAD`, `ADD`, `SUB`, `STORE`, `OUT` and `HALT`. During each fetch, print `fetch <MAR>: <instruction>` (for example `fetch 0: LOAD 6`). The program outputs `output: 35`.

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

memory = ["LOAD 6", "SUB 7", "ADD 8", "STORE 9", "OUT 9", "HALT", 40, 12, 7, 0]
pc = mar = acc = 0
mdr = None
```

The hint students can ask for: Each time round: copy the program counter into the address register, fetch what is there, report it, then move the counter on. Decode and carry out the instruction after that, as the lesson's version does.

A solution

```
from bugbot import *
connect()
memory = ["LOAD 6", "SUB 7", "ADD 8", "STORE 9", "OUT 9", "HALT", 40, 12, 7, 0]
pc = mar = acc = 0
mdr = None
while True:
    mar = pc
    mdr = memory[mar]
    print(f"fetch {mar}: {mdr}")
    pc = pc + 1
    op, *arg = mdr.split()
    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
```

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