ANSWERS

F9.5 CPU performance

Logic and computer systems · GCSE · OCR J277 1.1.2, AQA 8525 3.4.5 · about 15 min

BugBotLab

What this lesson is about

Clock speed, cores and cache, measured with models.

- Clock speed
- Cores
- Cache
- Putting it together

Questions

5 marks in all

1. What does a clock speed of 3 GHz mean? [1 mark]

- ☐ **A** 3 billion clock pulses each second
- ☐ **B** 3 million instructions in total
- ☐ **C** 3 cores
- ☐ **D** 3 GB of cache

> **Answer:** A. Hertz is per second; giga is a billion.

2. Why might doubling the cores not halve the time a program takes? [1 mark]

- ☐ **A** Some of the work cannot be split across cores
- ☐ **B** Cores share one clock pulse
- ☐ **C** More cores use less cache
- ☐ **D** The operating system can only use one core

> **Answer:** A. Steps that depend on the last answer must run one after another.

3. Why does more cache usually make a CPU faster? [1 mark]

- ☐ **A** More data is kept in fast memory, so fewer slow trips to RAM are needed
- ☐ **B** Cache increases the clock speed
- ☐ **C** Cache adds more cores
- ☐ **D** Cache stores files when the power is off

> **Answer:** A. A cache hit avoids fetching from slower main memory.

4. A job takes 60 seconds on 1 core and all of it can be split. How many seconds on 4 cores? [1 mark]

> **Answer:** 15. 60 ÷ 4.

5. What is a drawback of a higher clock speed? [1 mark]

- ☐ **A** The processor gets hotter and uses more power
- ☐ **B** It needs more RAM
- ☐ **C** It makes cache slower
- ☐ **D** It reduces the number of cores

> **Answer:** A. Heat and battery life limit clock speed.

The task: the cache model

Use `run_cache(requests, size)` on the request list `[5, 6, 5, 7, 5, 6, 8, 5, 6, 7]`, for cache sizes 1 to 6. Print one line for each size in the form `size <n>: <hits> hits`, then print `best size: <n>` for the smallest cache that gets the most hits, working it out with a loop.

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

def run_cache(requests, size):
    cache = []
    hits = 0
    for address in requests:
        if address in cache:
            hits = hits + 1
            cache.remove(address)
        elif len(cache) == size:
            cache.pop(0)
        cache.append(address)
    return hits, len(requests) - hits

requests = [5, 6, 5, 7, 5, 6, 8, 5, 6, 7]
```

The hint students can ask for: Call the model once for each size in the range, printing as you go. Track the best size by keeping it only when a size beats the best hits so far, so the smallest of equally good sizes wins.

A solution

```
from bugbot import *
connect()
def run_cache(requests, size):
    cache = []
    hits = 0
    for address in requests:
        if address in cache:
            hits = hits + 1
            cache.remove(address)
        elif len(cache) == size:
            cache.pop(0)
        cache.append(address)
    return hits, len(requests) - hits

requests = [5, 6, 5, 7, 5, 6, 8, 5, 6, 7]
best = 1
best_hits = -1
for size in range(1, 7):
    hits, misses = run_cache(requests, size)
    print(f"size {size}: {hits} hits")
    if hits > best_hits:
        best = size
        best_hits = hits
print("best size:", best)
```

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