ANSWERS

F8.4 Binary addition, overflow and shifts

Data representation · GCSE · OCR J277 1.2.4, AQA 8525 3.3.4, Edexcel 1CP2 2.1.4 · about 15 min

What this lesson is about

Adding in binary with carries, overflow, and shifting to multiply and divide.

- Adding binary
- Overflow
- Shifts

Questions

5 marks in all

1. Add 01011010 and 00110111 in binary. [1 mark]

> **Answer:** 10010001. 90 + 55 = 145, which is 10010001.

2. What is overflow? [1 mark]

- ☐ **A** A result too large to fit in the number of bits available
- ☐ **B** A number with too many decimal places
- ☐ **C** Running out of memory for files
- ☐ **D** A loop that never ends

> **Answer:** A. The extra bit is lost, so the stored answer is wrong.

3. In 8 bits, 200 + 100 overflows. What value do the 8 bits kept hold? [1 mark]

> **Answer:** 44. 300 - 256 = 44.

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

```
print(13 << 1, 13 << 2, 13 >> 1)
```

> **Answer:**
>
> ```
> 26 52 6
> ```
>
> Left shifts multiply by 2 and 4; a right shift halves, losing the remainder.

5. What does shifting a binary number two places left do? [1 mark]

- ☐ **A** Multiplies it by 4
- ☐ **B** Multiplies it by 2
- ☐ **C** Divides it by 4
- ☐ **D** Adds 2

> **Answer:** A. Each place left doubles: two places is × 4.

The task: an 8-bit adder

Write `add_binary(a, b)` yourself (no `int(..., 2)` or `bin` inside it) that adds two 8-bit strings and returns the 8-bit answer and the carry out. Print `01011010 + 00110111 = 10010001` and, on the next line, `11001000 + 01100100 = 00101100 overflow`, working both answers out with your function.

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

def add_binary(a, b):
    return "00000000", 0
```

The hint students can ask for: Work from the rightmost column to the left, adding the two bits and the carry from the column before. The column's answer is what is left after taking out any twos, and the new carry is whether there was a two. Build the answer up in front of what you have so far.

A solution

```
from bugbot import *
connect()
def add_binary(a, b):
    result = ""
    carry = 0
    for i in range(7, -1, -1):
        total = int(a[i]) + int(b[i]) + carry
        result = str(total % 2) + result
        carry = total // 2
    return result, carry

for a, b in [("01011010", "00110111"), ("11001000", "01100100")]:
    answer, carry = add_binary(a, b)
    print(a, "+", b, "=", answer + (" overflow" if carry else ""))
```

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