ANSWERS

F8.5 Negative numbers: two's complement

Data representation · GCSE · Edexcel 1CP2 2.1.2 · about 12 min

What this lesson is about

Signed 8-bit integers, and why the same adder works for them.

- The idea
- Making a number negative
- Why it works so well
- Negative speeds

Questions

4 marks in all

1. Write -5 in 8-bit two's complement. [1 mark]

> **Answer:** 11111011. 5 is 00000101; flip to 11111010; add 1.

2. What is 11111111 in 8-bit two's complement? [1 mark]

> **Answer:** -1. -128 + 127 = -1.

3. What range can an 8-bit two's complement number hold? [1 mark]

- ☐ **A** -128 to 127
- ☐ **B** -127 to 127
- ☐ **C** 0 to 255
- ☐ **D** -255 to 255

> **Answer:** A. The leftmost bit is worth -128.

4. What does a leftmost bit of 1 mean in two's complement? [1 mark]

- ☐ **A** The number is negative
- ☐ **B** The number is odd
- ☐ **C** The number is over 100
- ☐ **D** Overflow happened

> **Answer:** A. Only the -128 place can make the total negative.

The task: signed readings

Write `to_twos(n)` yourself for -128 to 127 (you may use `format(n, "08b")` for the positive part). Print the 8-bit two's complement of 42, -42 and -128, one per line, in the form `-42 = 11010110`.

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

def to_twos(n):
    return format(n, "08b")
```

The hint students can ask for: For a negative n: write -n in 8-bit binary, flip every bit, and add 1.

A solution

```
from bugbot import *
connect()
def to_twos(n):
    if n >= 0:
        return format(n, "08b")
    flipped = ""
    for bit in format(-n, "08b"):
        flipped = flipped + ("0" if bit == "1" else "1")
    return format(int(flipped, 2) + 1, "08b")

for n in [42, -42, -128]:
    print(n, "=", to_twos(n))
```

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