WORKSHEET

A9.6 AQA assembly language and bitwise operations

Computer architecture · A level · OCR H446 1.2.4, AQA 7517 4.7.3.5 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

AQA's instruction set, compare and branch, masks and shifts, and the bit fields that drive BugBot's motors.

- The instruction set
- Selection
- Iteration
- Running AQA assembly in Python
- What the bitwise operations are for
- Bits that drive motors

Questions

6 marks in all

1. What is the difference between MOV R1, #50 and LDR R1, 50? [1 mark]

- ☐ **A** MOV puts the number 50 in R1; LDR loads the value stored at address 50
- ☐ **B** They do the same thing
- ☐ **C** MOV loads from address 50; LDR puts the number 50 in R1
- ☐ **D** LDR moves R1 to address 50

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

```
r0 = 0b11001010
print(format(r0 & 0b00001111, '08b'))
print(format(r0 | 0b00000001, '08b'))
print(format(r0 >> 3, '08b'))
```

3. R0 holds 13. What does R1 hold after LSL R1, R0, #2? [1 mark]

4. Which instruction toggles (flips) the lowest four bits of R2, leaving the others alone? [1 mark]

- ☐ **A** EOR R2, R2, #15
- ☐ **B** AND R2, R2, #15
- ☐ **C** ORR R2, R2, #15
- ☐ **D** MVN R2, R2

5. A program does CMP R0, #10 then BLT small. When does it branch? [1 mark]

- ☐ **A** When the value in R0 is less than 10
- ☐ **B** When 10 is less than the value in R0
- ☐ **C** When R0 holds 10
- ☐ **D** Always

6. Why does an if-else in AQA assembly need an unconditional B after the 'if' part? [1 mark]

- ☐ **A** Otherwise the processor falls through into the else part as well
- ☐ **B** B stops the program
- ☐ **C** Branch instructions must come in pairs
- ☐ **D** It resets the flags

The task: decode the control bytes

`commands` holds four motor control bytes, each a whole number from 0 to 255 laid out as in the table above. For each byte, use bitwise operators (`>>` and `&`) to split it into its speed (0 to 63) and its mode, and: - print a line in the form `161: speed 40 forward` : the byte in denary, the speed, then the mode as `coast` , `forward` , `reverse` or `brake` ; - then act on it: `forward` drives forward 10 cm at that speed, `reverse` drives backward 10 cm at that speed, `brake` calls `stop()` , and `coast` does nothing. Do not convert the byte to a string of bits.

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

commands = [161, 98, 3, 0]
modes = ["coast", "forward", "reverse", "brake"]
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a9-6-aqa-assembly-and-bitwise-operations/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Write AQA assembly that sets bit 0 of the value at address 60 to 1 and stores it back, leaving the other bits alone.
2. Write AQA assembly that multiplies the value in R0 by 10 using only shifts and one ADD. (Hint: 10 = 8 + 2.)
3. Write AQA assembly that counts how many of the bits in R0 are 1.