WORKSHEET

A9.2 The processor and its registers

Computer architecture · A level · OCR H446 1.1.1, AQA 7517 4.7.3.1, Eduqas A500QS 2.1 · about 20 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

The ALU, control unit and clock, general-purpose and dedicated registers, and the status flags an 8-bit ALU sets.

- The components of the processor
- General-purpose and dedicated registers
- The status register's flags
- How flags make decisions
- Registers, cache and memory

Questions

5 marks in all

1. Which register holds the instruction currently being decoded and executed? [1 mark]

- ☐ **A** Current instruction register
- ☐ **B** Program counter
- ☐ **C** Memory address register
- ☐ **D** Status register

2. What does the control unit do? [1 mark]

- ☐ **A** Decodes instructions and sends control signals to coordinate the other components
- ☐ **B** Performs arithmetic and logic operations
- ☐ **C** Stores the result of calculations
- ☐ **D** Generates the clock pulses

3. An 8-bit ALU adds 200 and 100. Which flag is set? [1 mark]

- ☐ **A** Carry
- ☐ **B** Zero
- ☐ **C** Negative
- ☐ **D** Overflow

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

```
def flags(a, b):
    r = (a + b) & 255
    return r, r >> 7, int(r == 0)

print(flags(100, 50))
print(flags(128, 128))
```

5. What is the difference between a general-purpose register and a dedicated register? [1 mark]

- ☐ **A** A general-purpose register holds whatever the program chooses; a dedicated register has one fixed job, such as the PC
- ☐ **B** General-purpose registers are in main memory; dedicated ones are in the processor
- ☐ **C** Dedicated registers can only hold addresses
- ☐ **D** There is no difference

The task: the flags

Write `alu_add(a, b)` yourself for an 8-bit ALU. Its inputs `a` and `b` are whole numbers from 0 to 255 (bytes). It works out the 8-bit result (0 to 255, anything that does not fit is lost) and the four flags N, Z, C and V, each 0 or 1, exactly as the table above defines them. For each pair in `pairs`, print one line in the form `200 + 100 = 44 N=0 Z=0 C=1 V=0`: the two inputs, the 8-bit result, then the flags in the order N, Z, C, V. The robot does not move.

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

pairs = [(20, 30), (100, 50), (200, 100), (128, 128), (255, 1)]

def alu_add(a, b):
    return a + b
```

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

Do it on the robot
www.bugbotlab.com/learn/a9-2-the-processor-and-its-registers/
The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Write `alu_sub(a, b)` that works out `a - b` as an 8-bit subtraction and sets N, Z and V.
2. Find a pair of bytes that sets C but not V, and a pair that sets V but not C, without running the program. Then check.
3. Why does a processor need a CIR as well as an MDR, when both can hold the same instruction?