WORKSHEET

A9.10 Project: a processor of your own

Computer architecture · A level · OCR H446 1.1.1, AQA 7517 4.7.1.1, Eduqas A500QS 2.1 · about 35 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Build a von Neumann machine with register transfers, addressing modes, flags and memory-mapped I/O that plays a scale on the robot.

- The machine
- The program in memory
- Step 1: fetch
- Step 2: decode and the addressing modes
- Step 3: store and memory-mapped I/O
- Step 4: flags and branches
- Step 5: finish

Questions

4 marks in all

1. In the project machine, storing a value at address 255 plays a note. What is this technique called? [1 mark]

- ☐ **A** Memory-mapped I/O
- ☐ **B** Indirect addressing
- ☐ **C** Pipelining
- ☐ **D** Virtual storage

2. After CMP #700 with 650 in the accumulator, which flags are set in the project machine? [1 mark]

- ☐ **A** N = 1 and Z = 0
- ☐ **B** N = 0 and Z = 1
- ☐ **C** N = 0 and Z = 0
- ☐ **D** N = 1 and Z = 1

3. The loop in the project program is 6 instructions long and runs 4 times. With 4 instructions before it and HLT after it, how many fetches are there? [1 mark]

_____

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

```
memory = ['LDA #5', 'ADD 5', 'STA 6', 'HLT', 0, 7, 0]
pc = acc = 0
while True:
    cir = memory[pc]
    pc += 1
    op, _, x = cir.partition(' ')
    if op == 'HLT':
        break
    value = int(x[1:]) if x.startswith('#') else memory[int(x)]
    if op == 'LDA': acc = value
    elif op == 'ADD': acc += value
    elif op == 'STA': memory[int(x)] = acc
print(memory[6], pc)
```

The task: build the processor

Build the machine described above and run the program. The starter loads the program into memory and sets up the registers; you write everything else, using exactly the variable names `pc`, `mar`, `mdr`, `cir` and `acc` for the registers. - After every fetch, print `PC=<pc> MAR=<mar> CIR=<cir>` with the register values straight after the four fetch transfers, for example `PC=10 MAR=9 CIR=BLT 4`. - Storing to address 254 calls `forward(50, distance=<value>)`; storing to address 255 calls `tone(<value>, 0.2)`. - When `HLT` is fetched, stop and print `halted after <cycles> cycles`, counting every fetch including the `HLT`, then `ACC=<acc> Z=<z> N=<n>`. A working machine drives 10 cm, plays 300, 400, 500 and 600 Hz, prints 29 fetch lines, then `halted after 29 cycles` and `ACC=700 Z=1 N=0`.

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

memory = [0] * 256
program = [
    "LDA #10", "STA 254", "LDA #300", "STA 20", "LDA 20", "STA 255",
    "ADD #100", "STA 20", "CMP #700", "BLT 4", "HLT",
]
memory[:len(program)] = program

pc, mar, mdr, cir, acc = 0, 0, 0, "", 0
z, n = 0, 0
cycles = 0
```

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

Do it on the robot

www.bugbotlab.com/learn/a9-10-project-a-processor-of-your-own/

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

Challenges

1. Write a new program for your machine, in its memory, that drives forward 5 cm three times using a counter stored in memory and `BEQ`.
2. Add an interrupt: a list of cycle numbers at which a timer fires. At the end of those cycles, push the registers onto a stack, run a short ISR stored elsewhere in memory that plays a high note, and pop them back. Show that the scale still plays correctly.
3. Add an `LDX` instruction and indexed addressing, and use it to play a tune stored as a list of notes in memory from address 40.
4. Count how many memory reads your machine does in one run. Which instructions need the most, and which addressing mode would cut them?