IB Computer Design - Supervision 2

Nandor Licker <nl364@cl.cam.ac.uk>

Due noon before the day of supervision

Q1 2018 Paper 5 Question 1

Q2 2017 Paper 5 Question 1

Q3 2013 Paper 5 Question 1

Q4 2014 Paper 5 Question 1

Q5 You are to design a very simple computer, with a single register and 256 bytes of memory. The list of instructions you should implement follows:

- `load` *addr*: Loads a byte from memory into the accumulator.
- `store` *addr*: Stores a byte into memory from the accumulator.
- `jz` *addr*: Jumps to an address if the accumulator is 0.
- `imm` *imm*: Loads an immediate into the accumulator.
- `sub` *addr*: Subtracts the value located at *addr* from the accumulator.
- `add` *addr*: Adds the value located at *addr* to the accumulator.
- `in`: Inputs a byte from an external port.
- `stop`: Stops the CPU.

1. Write a program to compute the nth Fibonacci number, where n is read using the `in` instruction. The result should be stored in the accumulator on exit. The sequence starts at 1: 1, 1, 2, 3, ...
2. How many bytes are required to encode individual instructions? Can you pack the `imm` instruction into a single byte? Can you modify the `jmp` instruction to pack it into a single byte? What are the tradeoffs? Show the encoding of all instructions (do not compress them).
3. Encode your Fibonacci program.
4. How many cycles are required to execute each instruction? What stages do you need to break down each instruction to (instruction fetch, decode, operand fetch, writeback etc)? Draw a state machine (as a graph) where the nodes represent a stage (work executed in a single cycle) and the edges the instructions that are being executed. Do all instructions execute in the same number of cycles? For example, the sub instruction would be broken down into the following steps:
   - `opcode <= mem[pc]`
     `pc <= pc + 1`
   - `addr <= mem[pc]`
     `pc <= pc + 1`
   - `operand <= mem[addr]`
   - `add <= acc - operand`
5. Implement the computer in SystemVerilog. Use the Fibonacci example as a testbench. Do not forget the reset signal to set the processor to a sensible state.
6. Draw a diagram of the computer, indicating memory, ALU, any multiplexers, microcode, etc.