# Notes on computer architecture

- Hazards
  - Data hazards
    - In a pipelined processor, an instruction may follow an instruction that changes the value of the register needed. However, the result may not get written to the register before it is accessed, which will lead to an incorrect out-of-date input for that instruction.
    - Solutions
      - Stalling: Wait until the result is fully written to the register
      - Bypassing: forward the executed data, so the stalled instruction does not have to wait as long
        - Increases complexity and IPC
        - You do not have to bypass all stages, maybe just EXE and WB
  - Control hazards
    - In a pipelined processor, you will not necessarily know the PC (program counter) of the next instruction until a certain stage, depending on the instruction.
    - Not always the instruction that comes after - branch and jump instructions are possible
    - Solutions
      - Stalling: Wait until the next PC is available by freezing earlier pipeline stages
      - Bypassing: When the PC is available, route it back. The next executed instruction will be at this PC.
      - Speculating: Guess that the next instruction will be the next one in the program. If wrong, then kill and restart at correct PC.
- In-order processor: performs instructions in order.
  - simpler
  - less power
- Out-of-order processor: instructions are executed in an order such that hazards and stalling are avoided
  - more complex
  - more power
- Semaphores
  - wait: waits for the variable to be greater than 0, then decrements variable by 1.
  - signal: increments variable by 1
  - can be used to allow a certain number of resources to be used simultaneously
  - also can be used to ensure that only 1 process does something at a time (lock)
  - Deadlock: two processes cannot continue due to both processes requiring the same two resources
- Cache coherence
  - Conditions
    - Only one cache at a time has write permission for an address
    - No cache can have a stale copy of the data after a write to the address has been performed

- Rules
  - Write propagation: writes become visible to all processors
    - Write-invalidate: all other cached copies become invalid on write
    - Write-update: write is also performed to all other cached copies
  - Write serialization: writes are seen by all processors in the same order
    - Snooping: All caches observe each other's actions through shared bus
    - Directory: Coherence directory tracks contents of private caches, serializes requests.