YOUR NAME HERE: _____

CS 520 Quiz 1

Question 1: What is the Sequential Execution Model? Why is it important?

The sequential execution model specified that the CPU must execute instructions so that they appear to have been processed in the original program sequence. Most modern CPUs conform to this model, because it matches programmer intuition irrespective of implementation details of the microarchitecture.

By the way, what is the difference between the ISA and the microarchitecture?
The ISA is the contract between the compiler and the hardware, specifying the semantics of the instruction set (the language the hardware understands). The microarchitecture is the physical hardware implementation that conforms to the ISA.

Question 2: What is the difference between Dispatch and Issue? What physical component in an out-of-order CPU sits between those?

Dispatch is moving a decoded instruction into the Issue Queue. Issue is moving an instruction whose dependencies are resolved into an execution unit. The Issue Queue sits between dispatch and issue.

Question 3: What is the difference between instruction-level parallelism and machine parallelism?

Instruction-level parallelism (ILP) is a common characteristic of code – not all consecutive instructions are dependent on each other, so they could conceivable be executed at the same time or out of order.

Machine parallelism is any mechanism that takes advantage of the presence of ILP. Examples:
(1) The Issue Queue allows instructions to start up simultaneously or out of order.
(2) Having separate functional units for multiply, simple arithmetic, and memory access allows them to be processed in parallel.

Question 4: Name three causes of pipeline stalls.

- Dependencies (particularly dependencies upon long-latency instructions)
- Instruction cache misses
- Data cache misses
- Branches

Question 5: List the three types of instruction data dependencies. Indicate which two become a challenge in out-of-order CPUs. Why do they become a problem?

1. Flow/true/RAW dependencies – actual data flows between instructions.
2. Antidependencies/WAR – a later instruction writes to a register read by an earlier instruction.
3. Output/WAW dependencies – two instructions write to the same register.

2 and 3 are the false dependencies. If they are not respected, true dependencies elsewhere in the program may be interfered with.

Question 6: In a processor with an Issue Queue, by what mechanism are dependencies resolved?

Remember that "satisfying" a dependency is merely setting up for the data flow to occur at a later time, while "resolving" a dependency is the act of making the data flow actually happen. Commonly, dependency resolution occurs when a functional unit completes its work on an instruction and sends that result on a write-back bus. The register file and issue queue(s) observe this writeback, and if there are any issue queue entries dependent on that result, then the dependency will be resolved by storing the result in the issue queue. If all dependencies are resolved for an instruction in a queue, it will be issued to the appropriate functional unit.

Question 7: What is forwarding/bypassing?

Bypassing is any mechanism that allows the result of an instruction coming out of one functional unit to go directly to the input of another (or the same) functional unit for a dependent instruction. Example:

ADD R1,R2,R3
ADD R4,R1,R5

There is a flow dependency between these two instructions through R1. In the 5-stage pipeline, we can allow these two instructions to execute on consecutive cycles if we have a bypassing/forwarding mechanism that can route the ALU output back to its input.

Question 8: What is an interlock? What is the difference between hardware and software interlocking?

An interlock is any mechanism that detects a data hazard (some kind of dependency or resource conflict) and stalls the pipeline until the hazard has cleared. Hardware interlocks will detect hazards at runtime and stall the pipeline. Software interlocks will detect hazards at compile time and insert NOP instructions and statically reschedule instructions so as to avoid hazards.