CS230: Digital Logic Design and Computer Architecture
Tutorial 05, [Mon 23 Sep, Tue 24 Sep, Thu 26 Sep]

Concepts tested: Introduction to Pipelining, Structural Hazards, Pipelined Datapath, Data Hazards

1. Suppose we have the add3 instruction in MIPS, which uses the MEM cycle to do the extra addition, using the same ALU. Give an example of how this would cause a structural hazard. Give the corresponding pipeline timing diagram, with the necessary stalls.

2. Given the above stall(s), does add3 in the instruction set have any performance advantage? Explain.

3. Draw the pipeline timing diagram for the following two-instruction sequence, without any data forwarding, but assuming that the register file can be written in the first half of a cycle and read in the second half. Be sure to mark any repeated stages with a bubble.

   ```
   lw $t0, 4($sp)
   sw $t1, 8($t0)
   ```

4. Exhaustively list all the possible data forwarding required in the MIPS pipeline, to minimize data hazard related stalls. For each forwarding, mention the latch from which forwarding is done, and the stage to which forwarding is done.

5. **Zero offset memory access**

   (a) Many memory access instructions use zero as offset, with the usual base register. There is a possible advantage in having a special instruction for such cases. Say, we have a special instruction lw0, of the form:
   `lw0 $t0, ($sp) # equivalent to:  lw $t0, 0($sp)`
   Show a sequence of two instructions where, if a lw0 is used instead of the usual lw, a pipeline stall can be saved. You may assume that lw0 uses appropriate data forwarding in the pipelined MIPS implementation.

   (b) For your two-instruction sequence above, show the pipeline timing diagram for both cases: i.e. using lw, using lw0 instead of lw. Mark any stalls and data forwarding appropriately.

   (c) Show the datapath *change* required to implement data forwarding for the lw0 instruction above. Show *only the change*, not the entire pipelined datapath.