These concern a register-memory architecture that has both register-register and register-memory instructions, but no memory-memory ones. Addressing is always computed as (offset + base register). Arithmetic operations are

\[
\text{ALUop RegisterDestination, RegisterSource1, RegisterSource2 OR} \\
\text{ALUop RegisterDestination, RegisterSource1, MEM}
\]

and ALUop is in \{add, subtract, or, and, load, store\}. For load RegisterSource2 is ignored, for store RegisterDestination can be the same as RegisterSource1 or different – in which case RegisterSource1 is copied both to RegisterDestination and to MEM.

Branches do a comparison of two registers and branch to a PC-relative address determined by using the MEM field as an offset.

The pipeline used is

1. IF standard instruction fetch
2. RF instruction decode and register fetch
3. ALU1 effective address calculation for memory accesses and branches
4. MEM memory access
5. ALU2 operations and branch comparisons
6. WB writes back to register and/or memory

Register read and write on the same register can proceed in the same clock cycle.

1. How many adders (including incrementers and simple adders) does this pipeline need to avoid structural hazards for these units? Explain any relevant assumptions in your answer. Show a sequence of instructions that needs all of the adders that you have required.

2. How many register reads and writes can be required in parallel to avoid all structural hazards in accessing the registers? Explain any relevant assumptions in your answer. Show a sequence of instructions that needs all of the register ports that you have required.

3. In order to minimize data hazards, what forwarding should be done to the ALUs (assume that ALU1 and ALU2 run on different hardware). Remember to consider forwarding that must be preserved across more than one clock boundary.
4. In order to minimize data hazards, what other forwarding is needed to avoid stalls? In particular treat the case of memory references.

5. Enumerate all remaining data and control hazards, stating all assumptions clearly. Explain why (or if) you are confident that you have found them all. How many cycles must the instruction stall to resolve each hazard?

6. **598 only**: Propose at least two schemes to ensure that this pipeline has precise exceptions. Which do you prefer here, and why? (A short, cogent explanation is preferred to a long, rambling one.)

7. **598 only**: Suppose that the typical instruction mix contains 5% unconditional branches and 25% conditional branches, of which 40% are taken. What is the ratio of the time taken by the pipeline with appropriate stalls to that taken if no stalls were needed? Explain.

8. **598 only**: Using the data from Figure B.27 for the register-register MIPS architecture, give your most optimistic assessment (quantitatively!) of how this register-memory architecture might compare. Then give your most pessimistic assessment (quantitatively!).