Question 1. An instruction requires four stages to execute: stage 1 (instruction fetch) requires 30 ns, stage 2 (instruction decode) = 9 ns, stage 3 (instruction execute) = 20 ns and stage 4 (store results) = 10 ns. An instruction must proceed through the stages in sequence. What is the minimum asynchronous time for any single instruction to complete?

Question 2. We want to set this up as a pipelined operation. How many stages should we have and at what rate should we clock the pipeline?

Question 3. For the pipeline in question 2, how frequently can we initiate the execution of a new instruction, and what is the latency?

Question 4. What is the speedup of the pipeline in question 2?

Question 5. Draw the reduced state-diagram and show the maximum-rate cycle using the following collision vector:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Question 6. We have a RISC processor with register-register arithmetic instructions that have the format $R1 \leftarrow R2 \, \text{op} \, R3$. The pipeline for these instructions runs with a 100 MHz clock with the following stages: instruction fetch = 2 clocks, instruction decode = 1 clock, fetch operands = 1 clock, execute = 2 clocks, and store result = 1 clock.

a) At what rate (in MIPS) can we execute register-register instructions that have no data dependencies with other instructions?

b) At what rate can we execute the instructions when every instruction depends on the results of the previous instruction?

c) We implement internal forwarding. At what rate can we now execute the instructions when every instruction depends on the results of the previous instruction?
Question 7. Conditional branches are a problem with instruction pipelines. For the RISC processor described in question 6, we decide to implement branches by always assuming the branch will not be taken rather than implementing some form of branch prediction or speculative execution, and we do not implement internal forwarding. We don't know that the instruction is a branch until stage 2 (decode), we don't know the condition code setting (for instructions that set the condition code) until stage 5 (operand store) is complete, and we can't provide the target address (of a branch taken) to stage 1 until the end of stage 5. Assume a sequence of instructions where the condition code setting instruction immediately precedes the conditional branch.

a) What penalty in lost cycles do we incur for the branch not taken?

b) What penalty in lost cycles do we incur for the branch taken?

c) We implement delayed branching and the conditional branch is a delayed conditional branch. What penalty in lost cycles do we incur for the delayed branch taken?

d) We implement internal forwarding along with the delayed branch. What penalty in lost cycles do we incur for the delayed branch taken with internal forwarding?

Question 8. What is a greedy cycle?

Question 9. Why would you implement a branch history table in a pipelined computer?

Question 10. What do we mean when we say a computer is superscalar?

Question 11. What problem is speculative execution trying to solve?