1. Define type of hazards in RISC pipeline architecture, and how to solve them?

- Structural hazard (if trying to use the same resource at the same time from two separate parts of the pipeline). This hazard can be avoided by waiting, “stalling”, or with separate data and instruction memories.

- Data hazard (if trying to use data prematurely, for example using a command to a register which a previous pipeline is still operating on). Can be avoided by stalling or forwarding the results immediately after execution to the levels in which they are needed in. The structure can also include a unit that recognizes the hazards and then stalls or forwards data.

- Control hazard (if trying to make commands before the results of comparing exist). Can be avoided by stalling, which has a large impact on CPI.

2. Spot all data dependencies (including ones that do not lead to stalls). Draw arrows from the stages where data is made available, directed to where it is needed. Circle the involved registers in the instructions. Assume no forwarding.

addi t0,t1,100 F D E M W #t0 and t1

lw t2,4($t0) F D E M W

add t3,t1,t2 F D E M W

sw t3,8(t0) F D E M W #t0 and t3

lw t5,0(t6) F D E M W

or t5,t0,t3 F D E M W #t3 and t5

3. Redraw the arrows for the above question assuming our hardware provides forwarding.

addi t0,t1,100 F D E M W #t0 and t1

lw t2,4($t0) F D E M W

add t3,t1,t2 F D E M W

sw t3,8(t0) F D E M W #t0 and t3

lw t5,0(t6) F D E M W

or t5,t0,t3 F D E M W #t3 and t5

4.

LW R1, 0(R2) F D E M W

ADDI R1, R1, 1 \* \* \* F D E M W

LW R3, 0(R4) F D E M W

SW R1, 0(R3) \* \* \* F D E M W

ADDI R2, R2, 4 F D E M W

SUB R5, R5, R4 F D E M W

BNEZ R5, Loop \* \* \* F D E M W