Review the lab sheet first, then respond to the problems below via Moodle.

Problem 1

Is it possible to jump execution to addresses higher than 2^{26} assuming a sufficiently large program memory and the current `PC` value is 0? Why or why not?

- (A) This is not possible, because the necessary hardware would conflict with the data memory bus.
- (B) This is not possible, because the address field in the `j` instruction is only 26 bits long.
- (C) This is possible, because `ori` and `lui` may be used to set up a jump target register for use with `jr`, which can access the entire 32 bit address space.
- (D) This is possible, because the branch predictor will infer the target code block automatically.
- (E) This is not possible, because the SRAM interface is only 26 bits wide.

Problem 2

Why is it useful to have a `jal` instruction, as opposed to `j`?

- (A) To provide a means by which the program counter may be read from a running program.
- (B) To provide a return address when calling procedures.
- (C) Register 31 would have been unused without the `jal` instruction.
- (D) So that the `PC` may be used as an operand during conditional branch operations.

Problem 3

Suppose you were writing a MIPS assembler program, and wished to perform a conditional branch if not equal (bne) to a label more than 2^{15} addresses away from the current `PC` value. How would you go about this?

- (A) This cannot be done, because `bne` can only use a signed 16 bit immediate value as it's branch target.
- (B) This cannot be done because the assembler will not allow conditional branches to labels more than 2^{15} addresses away.

- (C) This will never need to happen, because the assembler will automatically re-order code so that all conditional branch targets are within 2^{15} addresses of the `PC` value.
- (D) Place an unconditional jump (j) right after a beq, using the latter to skip the former if the operands are equal.
- (E) Perform an unconditional jump (j), but discard the result if the bne condition is not met.

Problem 4

Suppose you implemented the design described in the lab sheet, but omitted the `stall_EX` signal entirely. What would happen when performing a jump or (taken) branch?

- (A) The instruction immediately after the jump/branch would be executed, potentially putting the register file and control logic in an inconsistent state.
- (B) The design would continue to operate as intended, the `stall_EX` just makes it easier to read the waveforms in ModelSim.
- (C) Execution would split into two separate threads, and two separate results would be computed, albeit half as fast.
- (D) The jump/branch target address would be executed as an instruction, rather than updating the `PC`.