1. At some point in the execution of a program, registers R4, R6, and R7 contain the values 2400, 8000, and 3300 (all decimal) respectively. Show the contents of registers RA, RB, RZ, RY, and R6 in Figure 5.8 (below) at the end of the decode, compute, memory, and store steps as the computer executes "Add, R6 8000,R7 3300,R4 2400".
   1. Decode:
      1. RA = 3300
      2. RB = 2400
      3. RZ
      4. RY
      5. R6 =8000
   2. Compute:
      1. RA = 3300
      2. RB = 2400
      3. RZ = 5700
      4. RY
      5. R6= 8000
   3. Memory:
      1. RA = 3300
      2. RB = 2400
      3. RZ = 5700
      4. RY = 5700
      5. R6= 8000
   4. Store:
      1. RA = 3300
      2. RB = 2400
      3. RZ = 5700
      4. RY = 5700
      5. R6= 5700
2. Consider an instruction set in which the instruction encoding is such that register addresses for different instructions are not always in the same bit locations. What effect would that have on the execution steps of the instructions? What could you do to maintain the five-step execution sequence? Assume the hardware is similar to Figure 5.8.
   1. I think the reading of the instruction would have to change. The CPU expects certain registers to be in specific places depending on the instruction format, so when that format is not met, there would need to be some other system of relaying what is what for the instruction. This would probably mean you need another step to decipher order of an instruction before decoding, if the order is going to be arbitrary.
3. Why would memory indirect addressing (i.e., the instruction contains a memory address A, which in turn is a pointer to memory location B which contains the value) be difficult to implement on the hardware in Figure 5.8?
   1. Indirect addressing would be hard to implement in 5.8 because not all areas have access to memory and no temporary registers. Although, the book mentions that the offset of X needs to be 0 for indirect addressing through Ri (157).
4. What would happen if we omit MFC from the circuits that create the control signals for the fetch step?
   1. What would happen is there would be no signal telling that fetching is complete, so anything that takes more than one clock cycle to fetch (going to memory instead of cache) would not be fetched and could be interrupted/lost.
5. Can we implement memory indirect addressing on the hardware of Figure 5.24? If so, what steps would be necessary to accomplish it?
   1. Yes, but I am not exactly sure how. I know indirect addressing is basically using a pointer in memory to point to somewhere else, so I suspect that one would need to have an extra portion in the fetch to get the actual address the first one points to. Also. Since most portions of the hardware have access to memory, it could increment the address in memory as it progresses more easily.