1. List three characteristics of RISC ISAs
   1. Regularity -- inst size, op count, work/inst
   2. Simplicity -- relatively few instruction formats and insts.
   3. Register-memory architecture (arithmetic ops only work on registers. LD and ST don’t do computation, but move data to and from memory.
   4. Orthogonality -- no special purpose registers. All registers can be used for any purpose. All instructions behave similarly.

2. A colleague proposes adding ‘add3’ to your ISA: add3 r1, r2, r3, r4 => R[r1] = R[r2] + R[r3] + R[r4]. What changes will this require in the instruction encoding? Do you support their suggestion? Why or why not?
   1. changes: new inst format, fewer registers (because fewer bits/operand), need to add sub3, etc. too or break orthogonality.
   2. Take a stand and provide a rationale. My inclination is no, because add3 is probably rarely used and breaks regularity/orthogonality and reduces reg count. Arguements in favor: If add3 is very frequently used, it might reduce instruction count and, therefore, reduce latency.

3. Give one advantage and one disadvantage of variable length instructions.
   1. Adv: More flexibility, increased readability, increased code-density
   2. disadv: irregularity. Difficult to decode.

4. How is the pace of class so far? If it’s too fast or slow, what would you add or remove?
   1. The large majority of people said it was fine.