1. Give the Amdahl’s Law equation for total speedup, \( S_{\text{tot}} \), in terms of the speedup, \( S \), that an optimization achieves on a fraction, \( x \), of a program.

2. Optimization A speeds up 10% of a program by 3x. Optimization B speeds up 30% of the program by 10x. What is the combined affect of A and B, assuming that they affect disjoint portions of the program.

4. Assume the following code executes repeatedly over the course of a program’s execution:

   ```c
   struct {
       Link * next;
       int data;
   } Link;

   bool Search(Link * next, int k)
   {
       while(next) {
           if (next->data == k) {
               return true
           }
       }
       return false
   }
   ```

   Would a stream buffer or a victim buffer added to the L1 DCache do the most to improve performance?

5. In the code in #4, what kind of branch predictor would do the best job of predicting the branch for the “if” conditional inside the loop if (a) Search() is called repeated on the same, very short (e.g., length = 3) list and (b) different, very long lists (e.g., 1000s of links).

6. You are designing processors at XYZ Technology Inc. The XYZ ISA is a beautiful, elegant RISC ISA. Most of the instructions in your ISA look like this:

   ```c
   add r1, r2, r3 // which means r1 = r2 + r3
   ```

   Your co-worker suggests adding the following instruction:

   ```c
   ldaddmul k(r4), r3, r2, r1 // which means r2= (mem[r1+k] + r3) *r4
   ```

   Because it would save him some time writing a function that handles an obscure corner case in some code he’s writing that will only execute every 35th Tuesday at 9:52am.

   Given four principles of RISC design that his suggestion violates.
7. Mondrian memory provides a feature that allows programmers to specify fine grain protection policies to prevent programmer errors from causing a system to crash. How could the authors use a Amdahl’s Law-style argument to motivate Mondrian memory as opposed to some other approach to reducing system crashes?

8. Mondrian memory and single address space operating systems both change the way that the computer implements memory protection, but they do so for very different reasons. Describe how the motivations for the two papers differ, and give an example of the resulting implementation that is a result of those differences.

9. The Cryptomaniac paper evaluated two ways of providing architectural support for cryptography – a completely new ISA and adding instructions to an existing ISA. Adding instructions to existing ISAs is more common in practice. Give two reasons why this is the case, and then give two disadvantages of adding to an existing ISA rather than starting from scratch.

10. Assume that assembly language instructions have the format “add <dst>, <src1>, <src2>”. Draw the true and false dependencies between the following instructions, labeling each with “R” for RAW, “WW” for WAW, or “WR” for “WAR” and the register involved.

   add r1, r2, r3
   sub r1, r1, r4
   sub r3, r5, r6

11. The processor you are building, the HAL 9000, is based on the HAL 8000, a machine that used Tomasulo’s algorithm to implement out-of-order execution for floating point operations (where each operation requires multiple cycles to execute). You would like to add OOO functionality for integer operations (which take just a single cycle) in the HAL 9000 as well. What changes are necessary to make Tomasulo’s algorithm effective for integer operations (i.e., to allow it to actually execute multiple instructions per cycle)?

12. Does Tomasulo’s algorithm reduce reduce instruction latency or increase instruction bandwidth? Why?

13. Consider a Tomasulo-based floating point unit in which all operations take two cycle to execute, there is one common data bus, and there are 3 functional units. Which would have a larger impact on performance: Doubling the number of functional units from 3 to 6 or doubling the number of busses from 1 to 2?

14. The WaveScalar discusses “dataflow locality” and remarks that conventional OOO do a lot of extra work in executing each instruction because they do not exploit dataflow locality. What is this extra work and what parts of an OOO processor perform it?
15. TRIPS executes blocks of instructions together, and those blocks may include if-then-else constructs that would turn into branches in conventional ISAs. This means that a TRIPS program contains fewer branches than an equivalent RISC ISA program. Does this make branch prediction easier or hardware for TRIPS?

16. WaveScalar doesn't include a branch predictor. Does branch prediction even make sense for WaveScalar? TRIPS has a branch predictor. What differences between TRIPS and WaveScalar makes branch prediction natural in TRIPS?