Problems 1 and 2 will be graded. There are 55 points on these problems. *Note: You must do all the problems, even the non-graded ones.* If you do not do some of them, half as many points as they are worth will be subtracted from your score on the graded problems.

**Problem 1.** *(30 points) [Tanenbaum, 5.1, 5.2, modified] (a) Design an expanding opcode to allow all the following to be encoded in a 36-bit instruction (i.e., tell which opcodes would be assigned to which instructions):

- 7 instructions with two 15-bit addresses and one 3-bit register number.
- 500 instructions with one 15-bit address and one 3-bit register number.
- 50 instructions with no addresses or registers.

(b) Is it possible to design an expanding opcode to allow the following to be encoded in a 12-bit instruction, assuming that a register number is 3 bits?

- 4 instructions with 3 registers
- 255 instructions with one register
- 16 instructions with zero registers

(c) Assume that an instruction consists of \( n \) bits and that a register number is \( r \) bits long. If there are \( k_3 \) instructions with three registers and \( k_1 \) instructions with one register, how many instructions can there be with zero registers?

**Problem 2.** *(25 points) [Hennessy & Patterson 2.6] Several researchers have suggested that adding a register-memory addressing mode to a load-store machine might be useful. The idea is to replace sequences of

\[
\text{LOAD R1,0(Rb)} \quad \text{ADD R2,R2,R1}
\]

By

\[
\text{ADD R2,0(Rb)}
\]

Assume the new instruction will cause the clock cycle to increase by 5%. Use the instruction frequencies for the gcc benchmark on the load-store machine from Figure 2.32 of the text. The new instruction affects only the clock cycle and not the CPI.

(a) What percentage of the loads must be eliminated for the machine with the new instruction to have at least the same percentage?

(b) Show a situation in a multiple instruction sequence where a load of R1 followed immediately by a use of R1 (with some type of opcode) could not be replaced by a single instruction of the form proposed, assuming the same opcode exists.
**Problem 3.** (25 points) [Hennessy & Patterson 2.15] A condition code is a bit of processor state updated each time certain ALU operation(s) execute to reflect some aspect of the execution. For example, a subtract instruction may set a bit if the result is negative and reset it for a positive result. A later operation can refer to this specific “result sign” condition code bit to glean information about the subtract result, provided no other instruction of the set that updates the result sign condition code has executed in the meantime. The concept of dedicated condition codes can be generalized to an array of general-purpose condition bits. An instruction is encoded to use any one of the general-purpose condition bits, as selected by the compiler. What are the advantages and disadvantages of a collection of general-purpose condition bits compared to those of dedicated condition codes (see Figure 2.21 in text)?

**Problem 4.** (20 points) [Hennessy & Patterson 2.15] The design of MIPS provides for 32 general-purpose registers and 32 floating-point registers. If registers are good, are more registers better? List and discuss as many trade-offs as you can that should be considered by ISA designers examining whether to, and how much to, increase the numbers of MIPS registers.