Question 1: Amdahl's Law

Suppose that a CPU can be enhanced so it can execute floating point instructions x times faster than a given implementation.

(a) What is the minimal relative frequency of floating point instructions in a given program that will yield an overall speedup of 1.7, as function of x?

(b) Suppose that the relative frequency of floating point instructions in an application program is 50%, how high need x to be in order to achieve an overall speedup of 1.25?

Complete your answer by plotting the overall speedup as a function of the relative frequency of floating point instructions, for this particular value of x.

Question 2: CPI

Suppose you have a machine with a cache and an instruction mix as follows. The data below assumes a perfect cache.

Type	Frequency	Clock Cycles Count
ALU ops	30%	1
Loads	30%	2
Stores	20%	2
Branches	20%	2

With an imperfect cache, measurements show that instructions fetches have a miss rate of 5%, load/stores which reference the memory have a hit rate of 10%, and that the miss penalty is 40 cycles.

(a) Find the CPI for each instruction type.

(b) Find the CPI for the overall machine.

(c) Find the how much faster is the machine with no cache misses.

Question 3: Tradeoffs

Suppose you are considering a design change to an instruction set. The current design is a pure load/store architecture like MIPS or DLX. The case we consider yields an instruction mix identical to that in the previous question. For the programs we consider, 20% of the ALU ops do not reuse one operand (out of two), 15% do not reuse either operand, and 10% do not reuse the result of an operation. Whenever an operand or a result is not reused (by a subsequent instruction), it is apparently wasteful to keep it in a register, since that register could be used by operand/results that are reused.

You, a member of the design team, propose adding register-memory ALU instructions that can operate on one or both source operands in memory, or alternatively can have destinations in memory (target operands). These new instructions would have a basic clock cycle count of 1 plus one for each memory operand. It is however apparent that the design is more complicated, with the consequences that:

(1) the clock cycle count for branches is increased by n without affecting the clock cycle time.
(2) or, alternatively, the clock cycle time is increased by x% without affecting the branch clock cycle count.

Assume the compiler is smart enough to take advantage of the new architecture whenever possible and ignore all other systems issues.

(a) Determine the maximum tolerable increase n in clock cycle count for the branches which yields a design which performs better than the original design (if n exists).

(b) Determine the maximum tolerable percent increase x in clock cycle time which yields a design which performs better than the original design (if x exists).

Discuss results.

Question 4: More Tradeoff

(A) Consider that the CPI of a machine is 2 when all memory references including instruction fetches are cache hits. The real machine has a cache design with misses for both instruction fetches and data accesses. The only data accesses are by Loads and Stores and these form 40% of all instructions. The miss penalty is 40 cc and the miss rate is 1% for the data cache for both read and writes. For the instruction cache the miss penalty is also 40 cc but the miss rate is 2.5%.
What is the speed-up factor for the ideal machine when all memory references are cache hits, relative to the real machine?

(B) Consider adding an enhancement to a machine that would increase the clock cycle time by 5% but would save 30% of the Load/Stores. The instruction mix indicates that 40% of instruction are Load/Stores. Is the enhancement worth being implemented supposing it is free of cost?