When I wrote this problem, I was envisioning an idealized Harvard architecture where instruction memory access is instant. I’ve provided ASPI and ACPI for both that and the Von Neumann architecture used in the in-class example.

1. (8 points) For each of the described program behaviors below, what is the Average Memory Access Time, Average Stalls Per Instruction, and Average CPI. In all cases, assume the following penalties:
   Branch Mispredict - 1 cycle; L1 Hit - 1 cycle; L2 Hit - 20 cycles; L2 Miss - 400 cycles
   a. 10% branches, 20% mispredicted. 30% Accesses, 3% miss L1 (hit L2), 2% miss L2
      AMAT: $1 \times 100\% + 20cc \times (3\%+2\%) + 400cc \times (2\%) = 10cc$
      ASPI (Harvard): $30\% \times (20cc \times (3\%+2\%) + 400cc \times (2\%)) + 1cc \times 10\% = 2.7cc$
      ASPI (Von Neumann): $130\% \times (20cc \times (3\%+2\%) + 400cc \times (2\%)) + 1cc \times 10\% = 11.7cc$
      ACPI = 3.7 or 12.7
   b. 5% branches, 50% mispredicted. 50% Accesses, 5% miss L1 (hit L2), 1% miss L2
      AMAT: $1 \times 100\% + 20cc \times (5\%+1\%) + 400cc \times (1\%) = 6.2cc$
      ASPI (Harvard): $50\% \times (20cc \times (5\%+1\%) + 400cc \times (1\%)) + 1cc \times 5\% = 2.63cc$
      ASPI (Von Neumann): $150\% \times (20cc \times (5\%+1\%) + 400cc \times (1\%)) + 1cc \times 5\% = 7.83cc$
      ACPI = 3.63 or 8.83
   c. 30% branches, 10% mispredicted. 50% Accesses, 5% miss L1 (hit L2), 4% miss L2
      AMAT: $1 \times 100\% + 20cc \times (5\%+1\%) + 400cc \times (1\%) = 18.8cc$
      ASPI (Harvard): $50\% \times (20cc \times (5\%+4\%) + 400cc \times (4\%)) + 1cc \times 30\% = 8.93cc$
      ASPI (Von Neumann): $150\% \times (20cc \times (5\%+4\%) + 400cc \times (4\%)) + 1cc \times 30\% = 26.73cc$
      ACPI = 9.93 or 27.73

2. Consider changing a cache from direct-mapped to 2-way set associative without otherwise altering it.
   a. (2 points) Can this change increase the AMAT? Why or why not?
      Yes. Increasing the associativity will increase the complexity and thus access time. The savings in reducing cache misses may not make up for this increase.
   b. (2 points) If not, is the AMAT guaranteed to decrease? Why or why not?
      No. If the cache is not experiencing any conflict inefficiency, increasing the associativity will not increase the hit rate.
3. For the following question, assume the cache line is at least 16 bytes. Also assume that you’re using a compiler that doesn’t make any optimizations of this type.

You’re working on a program which randomly accesses elements in a large array of 12-byte structures. Every field in a structure is accessed before the program moves to another one.

Another programmer mentions hearing something about structures being powers of two, and suggests you add padding to the end of the structure to increase it to 16 bytes.

a. (4 points) Could this increase performance? Why?
   Yes. Since every field in the structure is accessed, we know that every cache line the structure overlaps will be accessed. With the 12-byte structure and 16-byte lines, 50% of array elements will lie in two cache lines. If we assume that the array is significantly larger than the cache. This would cause two misses per line instead of one.

b. (4 points) Could this decrease performance? Why?
   Yes. If the array is not aligned to a 16-byte boundary in memory, this would result in EVERY element crossing a cache line boundary. If the cache line is 32 bytes or larger then the portion of elements that cross cache lines drops to 25% or less. Increasing the structure size decreases the chance of two elements accesses with temporal locality also sharing spatial locality. Increasing the structure size will also increase bandwidth requirements from the lower cache level, which would further impact performance.

c. (5 points towards quiz 5) If instead the structure were 20 bytes, would padding it up to 32 be likely to affect performance in the same way as it did for the 12 to 16 change?
   No. With a 16-byte cache line, two lines would need to be accessed for either size. Increasing the structure to 32 bytes prevents any benefit from chance spatial locality. This increase would also much more significantly increase bandwidth requirements (60% vs 33%).