3.

(a) AMAT of I cache data read

（1） L1miss，从L2中取数据的时间 发生概率 0.02

(12+ (32*8)/128 / 0.266) = 19.52ns

(2) L2 miss, 从主存中取数据的时间 发生概率 0.02 * 0.15

(80 + (64*8)/128/0.133) = 110.06 ns

（3） L2 miss， dirty page write back，发生概率 0.02 * 0.15 * 0.5

(80 + (64*8)/128/0.133) = 110.06 ns

AMAT = 0.02 * 19.52 + 0.02 * 0.15 * 110.06 + 0.02 * 0.15 * 0.5 * 110.06 = 0.886 ns

(b)

AMAT of D cache data read

（1） L1miss，从L2中取数据的时间 发生概率 0.05

(12+ (16*8)/128 / 0.266) = 15.76ns

(2) L2 miss, 从主存中取数据的时间 发生概率 0.05 * 0.15

(80 + (64*8)/128/0.133) = 110.06 ns

（3） L2 miss， dirty page write back，发生概率 0.05 * 0.15 * 0.5

(80 + (64*8)/128/0.133) = 110.06 ns

AMAT = 0.05 * 15.76 + 0.05 * 0.15 * 110.06 + 0.05 * 0.15 * 0.5 * 110.06 = 2.026ns

(c)

AMAT of D cache data write

（1） L1写回L2，write through策略，但是write buffer消除了90%的write stall，所以发生stall 概率0.1

(12+ (16*8)/128 / 0.266) = 15.76ns

(2) L2 miss, 从主存中取数据的时间 发生概率 0.1* 0.15

(80 + (64*8)/128/0.133) = 110.06 ns

(3) L2 miss， dirty page write back，发生概率0.1* 0.15 * 0.5

(80 + (64*8)/128/0.133) = 110.06 ns

AMAT = 15.76 * 0.1 + 0.1 * 0.15 * 110.06 + 0.1 * 0.15 * 0.5 * 110.06 = 4.05ns

d

CPI = 1.35 + 0.886 + 0.2 * 2.026 + 0.1* 4*0.5 = 3.046

4.

a

single precision: 4B

64 B cache line: 16 elements

一次可以处理16*16的大小的block，有两个矩阵

所以cache size : 16 * 16 * 2 * 4 B = 2KB

b

如果是block version，对于每个cache line大小的一个单元，input矩阵读入1次，output矩阵写入1次，就好了

如果是unblock version，对于每一个cache line大小的一个单元，output的矩阵写入1次，但是input的时候要读入16次（每写一个元素到output都会读入一次）

所以比值是2:17（具体看我实验课讲这道题的时候，他的工作过程）

d

2-way associative，因为input矩阵和output矩阵中的每个cache line大小的单元有可能有相同的index，需要保证output读进来的时候不会把input给换出去

6 .

1.Reduce the miss penalty

——multilevel caches, critical word first, read miss before write miss, merging write buffers, and victim
caches

2. Reduce the miss rate

——larger block size, large cache size, higher associativity, way prediction and pseudo associativity,
and compiler optimizations

3.Reduce the miss penalty and miss rate via parallelism
——non blocking caches, hardware prefetching, and compiler prefetching

4.Reduce the time to hit in the cache
——small and simple caches, avoiding address translation, pipelined cache access, and trace caches