HW2 Solution

2.1

a.

64byte÷8byte=8 elements

(in metrix elements + out metrix elements) × byte per element
= (8×8+8×8) ×8Byte=**1Kbyte**

b.

The blocked version only has to fetch each input and output element once. The unblocked version will have one cache miss for every 64B/8B = 8 row elements. Each column requires 64Bx256 of storage, or 16KB. Thus, column elements will be replaced in the cache before they can be used again. Hence the unblocked version will have 9 misses (1 row and 8 columns) for every 2 in the blocked version.

c.

```
for (i = 0; i < 256; i=i+B) {
    for (j = 0; j < 256; j=j+B) {
        for(m=0; m<B; m++) {
            for(n=0; n<B; n++) {
                output[j+n][i+m] = input[i+m][j+n];
            }
        }
    }
}
```

d.

兩個點需要考慮：

1. operation執行的速度

   每4個cycle一次讀寫

2.
   a. cache fetch的速度 & b. cache conflict

      L2每2個cycle可以處理一個指令 & 資料需要16cycle才能傳完

   b. 選擇n way-set(題目要求)

若要兩個(in matrix & out matrix)互不干擾(conflict)的話，需要2way-set

write array需要足夠的時間prefetch→那如果in和out同時發要求呢? OK!→ 有4cycle可用!!

2way有128個區間，可以滿足(依次為0、15、31、47、63、79、95、111、127)

至於read array每個cycle讀一個值，只需要1way就夠了

e.

You should be able to determine the level-1 cache size by varying the block size. The ratio of the blocked and unblocked program speeds for arrays that do not fit in the cache in comparison to blocks that do is a function of the cache block size, whether the machine has out-of-order issue, and the bandwidth provided by the level-2 cache. You may have discrepancies if your machine has a write-through level-1 cache and the write buffer becomes a limiter of performance.

2.18 a. The average memory access time of the current (4-way 64 KB) cache is 1.69 ns. 64 KB direct mapped cache access time = 0.86 ns @ 0.5 ns cycle time = 2 cycles Way-predicted cache has cycle time and access time similar to direct mapped cache and miss rate similar to 4-way cache.

The AMAT of the way-predicted cache has three components: miss, hit with way prediction correct, and hit with way prediction mispredict: $0.0033 \times (20) + (0.80 \times 2 + (1 - 0.80) \times 3) \times (1 - 0.0033) = 2.26$ cycles = 1.13 ns.

b. The cycle time of the 64 KB 4-way cache is 0.83 ns, while the 64 KB direct-mapped cache can be accessed in 0.5 ns. This provides 0.83/0.5 = 1.66 or 66% faster cache access.

c. With 1 cycle way misprediction penalty, AMAT is 1.13 ns (as per part a), but with a 15 cycle misprediction penalty, the AMAT becomes: $0.0033 \times 20 + (0.80 \times 2 + (1 - 0.80) \times 15) \times (1 - 0.0033) = 4.65$ cycles or 2.3 ns.

d. The serial access is 2.4 ns/1.59 ns = 1.509 or 51% slower.

2.20 a. With critical word first, the miss service would require 120 cycles. Without critical word first, it would require 120 cycles for the first 16B and 16 cycles for each of the next 3 16B blocks, or $120 + (3 \times 16) = 168$ cycles.

b. It depends on the contribution to Average Memory Access Time (AMAT) of the level-1 and level-2 cache misses and the percent reduction in miss service times provided by critical word first and early restart. If the percentage reduction in miss service times provided by critical word first and early restart is roughly the same for both level-1 and level-2 miss service, then if level-1 misses contribute more to AMAT, critical word first would likely be more important for level-1 misses.

2.21 a. 16B, to match the level 2 data cache write path.

b. Assume merging write buffer entries are 16B wide. Because each store can write 8B, a merging write buffer entry would fill up in 2 cycles. The level-2 cache will take 4 cycles to write each entry. A nonmerging write buffer would take 4 cycles to write the 8B result of each store. This means the merging write buffer would be two times faster.

c. With blocking caches, the presence of misses effectively freezes progress made by the machine, so whether there are misses or not doesn't change the required number of write buffer entries. With nonblocking caches, writes can be processed from the write buffer during misses, which may mean fewer entries are needed.

2.22 In all three cases, the time to look up the L1 cache will be the same. What differs is the time spent servicing L1 misses. In case (a), that time = 100 (L1 misses) × 16 cycles + 10 (L2 misses) × 200 cycles = 3600 cycles. In case (b), that time = $100 \times 4 + 50 \times 16 + 10 \times 200 = 3200$ cycles. In case (c), that time = $100 \times 2 + 80 \times 8 + 40 \times 16 + 10 \times 200 = 3480$ cycles. The best design is case (b) with a 3-level cache. Going to a 2-level cache can result in many long L2 accesses (1600 cycles looking up L2). Going to a 4-level cache can result in many futile look-ups in each level of the hierarchy.

2.30 (a) The LRU policy essentially uses recency of touch to determine priority. A newly fetched block is inserted at the head of the priority list. When a block is touched, the block is immediately promoted to the head of the priority list. When a block must be evicted, we select the block that is currently at the tail of the priority list.

2.32

The cores will be executing 8 cores $\times$ 3 GHz/2.0CPI = 12 billion instructions per second. This will generate $12 \times 0.00667 = 80$ million level-2 misses per second. With the burst length of 8, this would be $80 \times 32\text{B} = 2560$ MB/s. If the memory bandwidth is sometimes 2X this, it would be 5120 MB/s. From Fig. 2.14, this is just barely within the bandwidth provided by DDR2-667 DIMMs, so just one memory channel would suffice.

2.40

Hibernating will be useful when the static energy saved in DRAM is at least equal to the energy required to copy from DRAM to Flash and then back to DRAM. DRAM dynamic energy to read/write is negligible compared to Flash and can be ignored.

$$Time = \frac{8 \times 10^9 \times 2 \times 2.56 \times 10^{-6}}{64 \times 1.6} = 400\, seconds$$

The factor 2 in the above equation is because to hibernate and wakeup, both Flash and DRAM have to be read and written once.