Computer Architecture
HW#2 Solution

Exercises

B.1

a.

(1-5%)×1+(5%)×105=**6.2 cycles**

b.

$\frac{64\text{KB}}{256\text{MB}}$ =0.00025

(0.00025) ×1+(1-0.00025)*105=**104.974 cycles**

c.

latency without cache : 100 cycles

cache 是為了在 data 有 locality 特性的時候，以少量的 memory access latency 為代價減少直接存取 mem 的次數

d.

(1-miss) ×(T-G)+(miss) ×(T+L) ≥T

miss ≥ $\frac{G}{G+L}$ =0.951923

當 miss rate>0.952，此時 cache 無用

B.3

	LRU	**FIFO**	**Random**
a	**104**	**100**	**100**
b	**104**	**101**	**100**
c	**44**	**40**	**40**
d	**24**	**21**	**20**
e	**26**	**25**	**25**
f	**50**	**45**	**45**
g	**30**	**26**	**25**

h.

LRU：(1-5%)×26+(5%-3%)×50+3%×30=**26.6 units**

FIFO：(1-5%)×25+(5%-3%)×45+3%×26=**25.43 units**

Random：(1-5%)×25+(5%-3%)×45+3%×25=**25.4 units**

B.4

a.

10+5×([$\frac{4}{8}$] -1) = **10 cycle**

b.

a line is 32 byte：10+5×([$\frac{32}{8}$] -1) = **25 cycle**

c.

total write 8 times, so：8×(10+5×([$\frac{4}{8}$] -1)) = **80 cycle**

d.

當同一個cache line有超過 **3** 次的寫入才被取代，使用write back會比較好

e.

Answers vary, but one scenario where the write-through cache will be superior
is when the line (all of whose words are updated) is replaced out and back
to the cache several times (due to cache contention) in between the different
updates. In the worst case if a the line is brought in and out eight times and a
single word is written each time, then the write-through cache will require
8 × 10 = 80 CPU cycles for writes, while the write-back cache will use 8 ×
25 = 200 CPU cycles.

要包含兩個點：1.某種情況　2.write through會比write back快

B.5

a.

L1←→L2：15ns+(32byte÷128bit) ×1÷266MHZ=15+2×3.75=22.5 ns

L2←→Mem：60ns+(64byte÷128bit) ×1÷133MHZ=60+4×7.5=90 ns

0.02×22.5+0.02×(1-0.8) ×90+0.02×(1-0.8) ×0.5×90=**0.99 ns**

b.

L1←→L2：15ns+1÷266MHZ=15+3.75=18.75 ns

L2←→Mem：60ns+(64byte÷128bit) ×1÷133MHZ=60+4×7.5=90 ns

0.05×18.75+0.05×(1-0.8) ×90+0.05×(1-0.8) ×0.5×90=**2.2875 ns**

c.

L1←→L2：15ns+1÷266MHZ=15+3.75=18.75 ns

L2←→Mem：60ns+(64byte÷128bit) ×1÷133MHZ=60+4×7.5=90 ns

0.05×18.75+0.05×(1-0.8) ×90+0.05×(1-0.8) ×0.5×90=**2.2875 ns**

d.

CPI+IF+DR+DW=0.7+(0.99+0.2×2.2875+0.05×2.2875) * 1.1 =2.4180625

Case Study

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.

2.2
Since the unblocked version is too large to fit in the cache, processing eight 8B elements requires fetching one 64B row cache block and 8 column cache blocks. Since each iteration requires 2 cycles without misses, prefetches can be initiated every 2 cycles, and the number of prefetches per iteration is more than one, the memory system will be completely saturated with prefetches. Because the latency of a prefetch is 16 cycles, and one will start every 2 cycles, 16/2 = 8 will be outstanding at a time.

Exercises

2.8
a.

direct：0.862540203138

2-way：1.12056746501

4-way：1.3713953042

b.

16KB：1.26889227474

32KB：1.34885567595

64KB：1.3713953042

c.

	Direct	2-way	4-way	8-way
miss/ins.	0.00664	0.00366	0.000987	0.000266
ref./ins	0.3	0.3	0.3	0.3
miss rate	0.022133	0.0122	0.00329	0.000887
access time	**0.8625402**	**1.1205675**	**1.3713953**	**2.0352199**
cycle time	**0.504086**	**0.5095251**	**0.8290499**	**0.7895929**
cycle hit	2	3	2	3
mem access time	10	10	10	10
cycle time	0.504086	0.5095251	0.8290499	0.7895929
miss penalty	20	20	13	13
avg. access cycle	2.3984	3.2074	2.03619	3.008867
avg. access time	1.209	1.634251	1.688103	2.37578

2.9

a. (1-0.00329) ×(0.8×2+0.2×3)+ 0.00329×20=2.258562 cycles=1.138509484332 ns

b. 0.8290499/0.504086=1.644659641 higher

c. (1-0.00329) ×(0.8×2+0.2×15)+ 0.00329×20=4.650666 cycles=2.344335621276 ns

d.

Cycle time	Aggressive	Conservative
Normal	1.753767	1.918825
Serial	2.409613	2.48130

2.409613/1.753767 = 1.373964 slower (or 2.48130/1.918825 = 1.293135 slower)

2.12

a. 16Byte, same as write bandwidth

b. merging：16Byte will fill in 2 cycle & 4 cycle to write→16Byte/4cycle

nin-merging：8Byte per 4 cycle→8Byte/4cycle →2 times speed up

c.

blocking：when miss, those misses are frozen in the write buffer until write finished.

non-blocking：we will need more entries in write buffers for it can be process in write buffer.

2.14

a.

This is similar to the scenario given in the figure, but tRCD and CL are both 5. In addition, we are fetching two times the data in the figure. Thus it requires 5 + 5 + 4 × 2 = 18 cycles of a 333MHz clock, or 18 × (1/333MHz) = 54.0ns.

b.

The read to an open bank requires 5 + 4 = 9 cycles of a 333MHz clock, or 27.0ns. In the case of a bank activate, this is 14 cycles, or 42.0ns. Including 20ns for miss processing on chip, this makes the two 42 + 20 = 62ns and 27.0 + 20 = 47ns. Including time on chip, the bank activate takes 62/47 = 1.32 or 32% longer.