5.1 (1) 32/8 = 4 bytes, 故 16/4 = 4 integers

(2) 我们使用大量 I, J, 故它的是.
对 I, 我们访问 B[I][0] 8000次, 故它也是

(3) 即 A[I][J], 因为 A[I][J] 紧靠于 A[I][J+1]

(4) 64000 + 64000 - 64 = 127936

127936 + 8 = 127944

故需 $\frac{127944}{4}$ = 31986 blocks

(5) 同(2), 为 I, J, B[I][0]

(6) 为 A[J][I]

5.2 (1) 如下图所示:

Word address	Binary Address	Tag	Index	Hit / Miss
3	0000 0011	0	3	M
180	1011 0100	11	4	M
43	0010 1011	2	11	M
2	0000 0010	0	2	M
191	1011 1111	11	15	M
88	0101 1000	5	8	M
190	1011 1110	11	14	M
14	0000 1110	0	14	M
181	1011 0101	11	5	M
44	0010 1100	2	12	M
186	1011 1010	11	10	M
253	1111 1101	15	13	M

(2) 如下图所示:

Word address	Binary Address	Tag	Index	Hit / Miss
3	0000 0011	0	1	M
180	1011 0100	11	2	M
43	0010 1011	2	5	M
2	0000 0010	0	1	H
191	1011 1111	11	7	M
88	0101 1000	5	4	M
190	1011 1110	11	7	H
14	0000 1110	0	7	M
181	1011 0101	11	2	H
44	0010 1100	2	6	M
186	1011 1010	11	5	M
253	1111 1101	15	6	M

(3) 如下图:

Word Address	Binary Address	Tag	Cache 1 Index	Cache 1 hit/miss	Cache 2 Index	Cache 2 hit/miss	Cache 3 Index	Cache 3 hit/miss
3	0000 0011	0	3	M	1	M	0	M
180	1011 0100	22	4	M	2	M	1	M
43	0010 1011	5	3	M	1	M	0	M
2	0000 0010	0	2	M	1	M	0	M
191	1011 1111	23	7	M	3	M	1	M
88	0101 1000	11	0	M	0	M	0	M
190	1011 1110	23	6	M	3	H	1	H
14	0000 1110	1	6	M	3	M	1	M
181	1011 0101	22	5	M	2	H	1	M
44	0010 1100	5	4	M	2	M	1	M
186	1011 1010	23	2	M	1	M	0	M
253	1111 1101	31	5	M	2	M	1	M

Cache 1 miss rate = 100%

Cache 1 total cycles = $12 \times 25 + 12 \times 2 = 324$

Cache 2 miss rate = $10/12 = 83\%$

Cache 2 total cycles = $10 \times 25 + 12 \times 3 = 286$

Cache 3 miss rate = $11/12 = 92\%$

Cache 3 total cycles = $11 \times 25 + 12 \times 5 = 335$

Cache 2 表现最好

(4) $32 \times 2^8 = 2^5 \times 2^8 = 2^{13}$

$\frac{2^{13}}{2} = 2^{12}$

$32 - 12 - 3 = 17$

$64 + 17 + 1 = 82$

$2^{12} \times 82 = 328 \times 2^{10} = 335872$ bits, 或 328 kibibits

又 $32 - n - 4 = 28 - n$

$128 + (28 - n) + 1 = 157 - n$

令 $2^n \times (157 - n) \geq 335872$

得 $n = 12$

故有 2^{12} blocks

(5)

例如，我们可以只使用以下引用：

0x00000000, 0x10000000, 0x00000000

第一个缓存将所有这些映射到第一个块，所以我们有3次未命中。

第二个缓存将所有这些映射到第一个集合。它将第一个地址上的数据保存为该集合的第一个元素，并将第二个地址上的数据保存为该集合的第二个元素（有2个未命中）。然而，第三个参考是一个打击！

(6)

这是可能的。它使用5位，所以25=32块，这是无效的，但它的工作。每一块都可以打。我要添加的唯一修改是我们使用更多的位，具体来说，10位——例如，块地址【31:22】异或块地址【21:12】。

5.3 (1) $2^5 = 32$ bytes $32/4 = 8$ words

(2) $2^5 = 32$

(3) $8 \times 32 = 256$ bits(8 words)

$22 + 1 + 256 = 279$ bits

比例: $\frac{256}{279} = 0.918 = 91.8\%$

(4)

Decimal	Binary	Offset	Index	Tag	Hit/Miss
0	0000 0000 0000	00000	00000	00	M
4	0000 0000 0100	00100	00000	00	H
16	0000 0001 0000	10000	00000	00	H
132	0000 1000 0100	00100	00100	00	M
232	0000 1110 1000	01000	00111	00	M
160	0000 1010 0000	00000	00101	00	M
1024	0100 0000 0000	00000	00000	01	M (replacement)
30	0000 0001 1110	11110	00000	00	M (replacement)
140	0000 1000 1100	01100	00100	00	H
3100	1100 0001 1100	11100	00000	11	M (replacement)
180	0000 1011 0100	10100	00101	00	H
2180	1000 1000 0100	00100	00100	10	M (replacement)

被替换 4个

(5) $\frac{4}{12} = \frac{1}{3} = 33.33\%$

(6) <Index, tag, data>

<000001$_2$, 0001$_2$, mem[1024]>

<000001$_2$, 0011$_2$, mem[16]>

<001011$_2$, 0000$_2$, mem[176]>

<001000$_2$, 0010$_2$, mem[2176]>

<001110$_2$, 0000$_2$, mem[224]>

<001010$_2$, 0000$_2$, mem[160]>