5.1

```
for(I = 0; I < 8; I++)
    for(J = 0; J < 8000; J++)
        A[I][J] = B[I][0] + A[J][I]
```

1. 16 / 4 = 4
2. $I, J, B[i][0]$
3. $A[i][j]$
4. 一个矩阵元素32位，即4字节，$A[i][j]$ 需要访问 8 × 8000 = 64000个元素，$B[i][0]$ 需要访问8个元素，一个cache块16字节，则需要$(64000 + 8) * 4/16 = 16002$ 个cache块
5. $I, J, B[i][0]$
6. $A[j][i]$

5.2.1

由于cache有16个块，块大小为1个字，则 32位地址中，0-1位表示块大小，2-5位表示索引，6-31位表示标记：

字地址	二进制地址	标记	索引	命中/缺失
3	0000 0011	0	3	缺失
180	1011 0100	11	4	缺失
43	0010 1011	2	11	缺失
2	0000 0010	0	2	缺失
191	1011 1111	11	15	缺失
88	0101 1000	5	8	缺失
190	1011 1110	11	14	缺失
14	0000 1110	0	14	缺失
181	1011 0101	11	5	缺失
44	0010 1100	2	12	缺失
186	1011 1010	11	10	缺失
253	1111 1101	15	13	缺失

其中二进制地址为字地址的二进制形式，共有30位，但是最大值为253，只用得到8位，因此前22位全是0，表格给出的是低8位

5.2.2

由于cache有8个块，块大小为2个字，则 32位地址中，0-2位表示块大小，3-5位表示索引，6-31位表示标记：

字地址	二进制地址	标记	索引	命中/缺失
3	0000 0011	0	1	缺失
180	1011 0100	11	2	缺失
43	0010 1011	2	5	缺失
2	0000 0010	0	1	**命中**
191	1011 1111	11	7	缺失
88	0101 1000	5	4	缺失
190	1011 1110	11	7	**命中**
14	0000 1110	0	7	缺失
181	1011 0101	11	2	**命中**
44	0010 1100	2	6	缺失
186	1011 1010	11	5	缺失
253	1111 1101	15	6	缺失

其中二进制地址为字地址的二进制形式，共有30位，但是最大值为253，只用得到8位，因此前22位全是0，表格给出的是低8位

5.3

1. $2^5/2^2=8$
2. 索引位总共有5位，则cache有 2^5 项
3. cache每一项数据存储 8 个字，共 $8\times4\times8=256$ 位，还需要标记 22 位， 则执行所需总位数与数据存储位数比为：

   $1+22/256=1.086$

字节地址	标记	索引	命中/缺失
0	0	0	缺失
4	0	0	**命中**
16	0	0	**命中**
132	0	4	缺失
232	0	7	缺失
160	0	5	缺失
1024	1	0	缺失，替换
30	0	0	缺失，替换
140	0	4	**命中**
3100	3	0	缺失，替换
180	0	5	**命中**
2180	2	4	缺失，替换

4. 按照上表，有4块被替换
5. 命中4次，命中率为 $4/12 = 33\%$
6.

索引	标记	数据地址
0	3	3072 - 3103
4	2	2176 - 2207
5	0	160-191
7	0	224-255

5.5.1

当地址 $32k$（k为整数）被访问时，会将 $[32k, 32k+31]$ 地址对应的数据全部载入缓存，即每16次访问，第1次会缺失，后面15次会命中，因此缺失率为 $\frac{1}{16}$;

cache 的缺失率与cahce大小以及工作集无关，与cache块大小有关;

每次缺失都是因为该块被第一次访问不在cache中，根据3C模型，是强制性失效(compulsory miss)

5.6

1. P1: $1/(0.66 * 10^{-9}s) = 1.52GHz$

   P2: $1/(0.90 * 10^{-9}s) = 1.11GHz$

2. 平均存储器访问时间 = 命中时间 + 缺失率 * 缺失代价

题目仅仅给出了访问主存的时间为 70ns，并没有交代缺失代价究竟是多少。若是70ns仅包括将主存内容传输至cache，那么还需要再次访问cache，因此P1缺失代价可认为是70.66ns, 若是70ns还包含了将数据直接传送给请求者的时间，那么P1缺失代价就是70ns

不妨就让P1缺失代价为70ns, P2类似处理

P1： $0.66 + 70 * 0.08 = 6.26\ ns = 9.48\ cycles$

P2： $0.90 + 70 * 0.06 = 5.10\ ns = 5.67\ cycles$

3. P1 CPI：$1 + 9.48 * 0.36 = 4.41$， 执行一条指令的时间为：$0.66 + 6.26 * 0.36 = 2.91\ ns$

   P2 CPI：$1 + 5.67 * 0.36 = 3.04$ 执行一条指令的时间为：$0.90 + 5.10 * 0.36 = 2.74\ ns$

   P2更快

4. P1：$0.66 + 0.08 * (5.62 + 0.95 * 70) = 6.43\ ns = 9.74\ cycles$

   有了二级cahce, AMAT反而更差

5. PI CPI：$1 + 9.74 * 0.36 = 4.51\ cycles$
6. 由题意可知，P1此时有了二级cache, AMAT为 6.43ns，P2未说明有没有增加二级cache, 若假设没有增加，则AMAT为5.10ns, 则还是P2处理器更快，假设P1的一级cache缺失率为 x 时，两个处理器的AMAT相同：

   可以得到方程： $0.66 + x * (5.62 + 0.95 * 70) = 5.10$， 解得 $x = 0.062 = 6.2\%$

5.11

1.在TLB中，标记位表示虚拟页地址，使用全相连方式，查找地址过程如下：

①若是再TLB表中(虚拟地址与标记位对应且有效位为1)，则找到，否则进入第②步；

②若是再PT中，如查找页面不在页表中，则发生异常；如找到，有效位为1，则该页在内存上，可以获取地址并按照LRU算法载入TLB，若有效位为0，则表示该页在磁盘中，需要先调入内存，而题目没有指明具体的调入方法，页表中最大物理页号为12，那么不妨假设磁盘调入的页从内存物理地址的第13页开始放置，再按照LRU算法载入TLB。

过程如下：

4669 // 4096 = 1， 虚拟页1，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页13，更新页表和TLB：

有效位	标记位	物理页号
1	11	12
1	7	4
1	3	6
1	1	13

2227 // 4096 = 0， 虚拟页0，不在TLB中，在页表中，对应物理页号5，更新TLB：

有效位	标记位	物理页号
1	0	5
1	7	4
1	3	6
1	1	13

13916 // 4096 = 3， 虚拟页3，在TLB中，无需更新；

34587 // 4096 = 8， 虚拟页8，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页14，更新页表和TLB：

有效位	标记位	物理页号
1	0	5
1	8	14
1	3	6
1	1	13

48870 // 4096 = 11， 虚拟页11,不在TLB中，在页表中，对应物理页号12，更新TLB：

有效位	标记位	物理页号
1	0	5
1	8	14
1	3	6
1	11	12

12608 // 4096 = 3， 虚拟页3，在TLB中，无需更新；

49225 // 4096 = 12， **超出页表索引范围，产生异常；**

最终TLB的状态：

有效位	标记位	物理页号
1	0	5
1	8	14
1	3	6
1	11	12

2.

4669 // 16384 = 0， 虚拟页0，不在TLB中，在页表中，对应物理页号5，更新TLB：

有效位	标记位	物理页号
1	11	12
1	7	4
1	3	6
1	0	5

2227 // 16384 = 0， 虚拟页0，在TLB中，无需更新；

13916 // 16384 = 0， 虚拟页0，在TLB中，无需更新；

34587 // 16384 = 2，虚拟页2，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页13，更新页表和TLB：

有效位	标记位	物理页号
1	2	13
1	7	4
1	3	6
1	0	5

48870 // 16384 = 2， 虚拟页2，在TLB中，无需更新；

12608 // 16384 = 0， 虚拟页0，在TLB中，无需更新；

49225 // 16384 = 3， 虚拟页3，在TLB中，无需更新；

最终TLB的状态：

有效位	标记位	物理页号
1	2	13
1	7	4
1	3	6
1	0	

使用更大的页，可以降低缺失率，提高存储器的访问时间，但是缺点是造成更多的内碎片，降低物理内存的使用效率。

3.

对于两路组相连，不妨假设TLB中第0，2项为第0组，第1,3项为第一组；

4669 // 4096 = 1， 虚拟页1，组号为1，标记位为0，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页13，更新页表和TLB：

有效位	组号	标记位	物理页号
1	0	11	12
1	1	7	4
1	0	3	6
1	1	0	13

2227 // 4096 = 0， 虚拟页为0，组号为0，标记位为0，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页14，更新页表和TLB：

有效位	组号	标记位	物理页号
1	0	0	5
1	1	7	4
1	0	3	6
1	1	0	13

13196 // 4096 = 3，虚拟页为3，组号为1，标记位为1，不在TLB中，在页表中，对应于物理页号6，更新TLB：

有效位	组号	标记位	物理页号
1	0	0	5
1	1	1	6
1	0	3	6
1	1	0	13

34587 // 4096 = 8，虚拟页为8，组号为0，标记位为4，不在TLB中，在页表中，发生缺页，从磁盘调入该页到主存的物理页15，更新页表和TLB：

有效位	组号	标记位	物理页号
1	0	0	5
1	1	1	6
1	0	4	14
1	1	0	13

48870 // 4096 = 11， 虚拟页为11，组号为1，标记位为5，不在TLB中，在页表中，对应于物理页号11，更新TLB：

有效位	组号	标记位	物理页号
1	0	0	5
1	1	1	6
1	0	4	14
1	1	5	12

12608 // 4096 = 3，虚拟页为3，组号为1，标记位为1，在TLB中，无需更新；

49225 // 4096 = 12，虚拟页为12，组号为0，标记位为6，不在TLB中，**超出页表范围，产生异常；**

最后的TLB表为：

有效位	组号	标记位	物理页号
1	0	0	5
1	1	1	6
1	0	4	14
1	1	5	12

对于直接映射：

4669 // 4096 = 1， 虚拟页1，对应索引为1，标记位为0，不在TLB表中，在页表中，发生缺页，从磁盘调入该页到主存的物理页13，更新页表和TLB：

有效位	标记位	物理页号
1	11	12
1	0	13
1	3	6
0	4	9

2227 // 4096 = 0，虚拟页0，对应索引为0，标记位为0，不在TLB表中，在页表中，对应于物理页5，更新TLB：

有效位	标记位	物理页号
1	0	5
1	0	13
1	3	6
0	4	9

13916 // 4096 = 3， 虚拟页3，对应索引3，标记位为0，不在TLB表中，在页表中，对应于物理页6，更新TLB：

有效位	标记位	物理页号
1	0	5
1	0	13
1	3	6
1	0	6

34587 // 4096 = 8，虚拟页8，对应索引0，标记位为2，不在TLB表中，在页表中，发生缺页，从磁盘调入该页到主存的物理页14，更新页表和TLB：

有效位	标记位	物理页号
1	2	14
1	0	13
1	3	6
1	0	6

48870 // 4096 = 11，虚拟页11，对应索引3，标记位为2，不在TLB中，在页表中，对应于物理页12，更新TLB：

有效位	标记位	物理页号
1	2	14
1	0	13
1	3	6
1	2	12

12608 // 4096 = 3，虚拟页3，对应索引3，标记位为0，不在TLB中，在页表中，对应于物理页6，更新TLB：

有效位	标记位	物理页号
1	2	14
1	0	13
1	3	6
1	0	6

49225 // 4096 = 12， 虚拟页为12，对应索引为0，标记位为3，不在TLB中，**超出页表范围，产生异常；**

最后的TLB表为：

有效位	标记位	物理页号
1	2	14
1	0	13
1	3	6
1	0	6

TLB可以快速地获取物理页与虚拟页的映射关系，在命中时，可以减少一次访问内存页表的时间，提高查找效率；若是没有TLB，那么每次查找页的地址都需要在页表中查找，访存时间会大大增加，使机器效率降低。