1. **Finish your homework independently**
2. **Convert this docx to pdf: “stuID_name_csapp4.pdf”**
   Example: ”202X010000_zhangsan_csapp4.pdf”
3. **Submit this pdf: learn.tsinghua.edu.cn**

6.12
In general, if the high-order s bits of an address are used as the set index, contiguous
chunks of memory blocks are mapped to the same cache set.
A. How many blocks are in each of these contiguous array chunks?

B. Consider the following code that runs on a system with a cache of the form
(S, E, B, m) = (512, 1, 32, 32):

```
int array[4096];
for (i = 0; i < 4096; i++)
sum += array[i];
```

What is the maximum number of array blocks that are stored in the cache
at any point in time?

A

解答：
在缓存系统中，集合的数量是由地址的高阶位决定的。对于系统配置(S, E, B, m) = (512, 1, 32, 32)，集合的数量S为512，意味着集合索引需要9位（2^9 = 512）。

由于缓存块的大小B为32字节，每个缓存块可以存储32字节的数据。在直接映射缓存中，每个集合只能存储一个缓存块。

每个连续的数组块中包含的内存块数量为1。

B
问题描述：

解答：
首先，数组`array[4096]`由4096个整数组成，假设整数大小为4字节，数组总大小为16384字节。由于缓存块大小B为32字节，因此数组被划分为16384 / 32 = 512个缓存块。

缓存有512个集合（S=512），并且是直接映射的，每个集合存储一个缓存块。但是，由于数组大小与缓存的总大小相同，并且缓存是直接映射的，访问数组的新块将替换掉当前缓存中的块。因此，尽管缓存可以容纳512个块，但实际上它在每次新的数组块被访问时都会发生替换。

在任何时间点，缓存中最多只能存储1个数组块。

6.24 ◆
Estimate the average time (in ms) to access a sector on the following disk:

Parameter	Value
Rotational rate	**12,000 RPM**
$T_{\text{avg seek}}$	**3 ms**
Average# sectors/track	**500**

- Rotational rate: 12,000 RPM (转速)
- T_(avgseek): 3 ms (平均寻道时间)
- Average # sectors/track: 500 (每磁道平均扇区数)

根据这些参数，可以计算平均访问时间，它包括三部分：

1. 平均寻道时间 (T_avgseek)
2. 平均旋转等待时间 (T_avgrotational)
3. 数据传输时间 (T_transfer)

由于只需要估计平均访问时间，而数据传输时间通常与寻道和旋转延迟相比很小，因此在这里我们可以忽略数据传输时间。

现在来计算平均旋转等待时间。磁盘每分钟旋转12,000次，因此每秒旋转200次(12,000 RPM / 60 seconds)。每次旋转的时间是1/200秒，即5毫秒。平均旋转等待时间是一次完整旋转时间的一半，即2.5毫秒。

现在可以加上平均寻道时间和平均旋转等待时间来得到总的平均访问时间：

- T_avgseek = 3 ms
- T_avgrotational = 2.5 ms
- T_transfer = Negligible (可以忽略)

将两者相加，可以得到：

T_avgaccess = T_avgseek + T_avgrotational = 3 ms + 2.5 ms = 5.5 ms

因此，访问磁盘上一个扇区的平均时间大约为5.5毫秒。

6.35 ◆◆

Consider the following matrix transpose routine:

```
typedef int array[4][4];

void transpose2(array dst, array src)
{
  int i, j;

  for (i = 0; i < 4; i++) {
    for (j = 0; j < 4; j++) {
      dst[i][j] = src[j][i];
    }
  }
}
```

Assume this code runs on a machine with the following properties:

. sizeof(int) == 4.

. The src array starts at address 0 and the dst array starts at address 64 (decimal).

. There is a single L1 data cache that is direct-mapped, write-through, writeallocate,
with a block size of 16 bytes.

. The cache has a total size of 32 data bytes and the cache is initially empty.

. Accesses to the src and dst arrays are the only sources of read and write misses, respectively.

A. For each row and col, indicate whether the access to src[row][col] and dst[row][col] is a hit (h) or a miss (m). For example, reading src[0][0] is a miss and writing dst[0][0] is also a miss.

	src array			
	Col 0	Col 1	Col 2	Col 3
Row 0	m	h	h	h
Row 1	m	h	h	h
Row 2	m	h	h	h
Row 3	m	h	h	h

	dst array			
	Col 0	Col 1	Col 2	Col 3
Row 0	m	m	m	m
Row 1	m	m	m	m
Row 2	m	m	m	m
Row 3	m	m	m	m

6.36 ◆◆
Repeat Problem 6.35 for a cache with a total size of 128 data bytes.

	src array			
	Col 0	Col 1	Col 2	Col 3
Row 0	m	h	h	h
Row 1	m	h	h	h
Row 2	m	h	h	h
Row 3	m	h	h	h

	dst array			
	Col 0	Col 1	Col 2	Col 3
Row 0	m	h	h	h
Row 1	m	h	h	h
Row 2	m	h	h	h
Row 3	m	h	h	h