1. **Finish your homework independently**
2. **Convert this docx to pdf: “stuID\_name\_csapp4.pdf”**

**Example: ”202X010000\_zhangsan\_csapp4.pdf”**

1. **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.**

1. **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?**

**6.24 ◆**

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

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| **Rotational rate** | **12,000 RPM** |
|  | **3 ms** |
| **Average# sectors/track** | **500** |

**6.35 ◆◆**

**Consider the following matrix transpose routine:**

**1 typedef int array[4][4];**

**2**

**3 void transpose2(array dst, array src)**

**4 {**

**5 int i, j;**

**6**

**7 for (i = 0; i < 4; i++) {**

**8 for (j = 0; j < 4; j++) {**

**9 dst[i][j] = src[j][i];**

**10 }**

**11 }**

**12 }**

**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** |  |  |  |
| **Row 1** |  |  |  |  |
| **Row 2** |  |  |  |  |
| **Row 3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **dst array** | | | |
| **Col 0** | **Col 1** | **Col 2** | **Col 3** |
| **Row 0** | **m** |  |  |  |
| **Row 1** |  |  |  |  |
| **Row 2** |  |  |  |  |
| **Row 3** |  |  |  |  |

**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** |  |  |  |  |
| **Row 1** |  |  |  |  |
| **Row 2** |  |  |  |  |
| **Row 3** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **dst array** | | | |
| **Col 0** | **Col 1** | **Col 2** | **Col 3** |
| **Row 0** |  |  |  |  |
| **Row 1** |  |  |  |  |
| **Row 2** |  |  |  |  |
| **Row 3** |  |  |  |  |