Advanced Computer Architecture CMSC 611
Homework 6
Due in class at 1.05pm, Dec 10th, 2012

(If you wish to **go green,** then you can submit the entire Homework electronically. Make sure you include the string **"CMSC 611 Homework"** in your subject line. You will get a canned response immediately if your message is filtered correctly! I use an exact substring match for the filter so make sure you include the exact string in your subject line. Deadline remains the same)

Please **DO NOT** email your homework to Dr. Olano!! **DO NOT** include him in the CC either!! There is a strong chance it won't be graded if you do!! **Send it only to <abhay1@umbc.edu>**

1) (20 points)

a) RAID 0 does not provide any redundancy to improve reliability. Instead it uses a technique called striping to improve disk performance. What exactly is striping and how does it improve I/O performance? Why might a person prefer to have 2 (or more) independent drives rather than have them in a RAID 0 setup?

Striping = storing different parts of a file across multiple disks. Improves performance because each disk can simultaneously read a different part of the file compared to a single disk reading the whole file.
RAID 0 provides no redundancy while spreading all your data across multiple disks so if you lose 1 disk, you lose all your data.

b) RAID3 and RAID 4 both provide extra disks for data redundancy. In what situations might one prefer to use RAID 4 over RAID 3? What is the bottleneck of RAID 4?

In RAID 3, every access goes to all disks, no matter how small. In RAID 4, smaller accesses are supported because not all disks need to be read for every file. The major bottleneck in RAID 4 is writes, because every single write needs to write to the parity disk.

c) What is the main difference between snooping protocols and directory protocols?

In snooping protocols, each cache only knows what it has and there is no central location to lookup information on the state of other caches. So all transactions are put on the bus and each cache must listen (snoop) the bus traffic to update hteir states accordingly.
In directory protocols, there is a global directory which caches which processors have what data.

d) Implementations of directory protocol where the whole directory is centralized in one location have problems scaling to more than a couple hundred processors. What is the

bottleneck in basic directory protocols and why does it occur? How might this bottleneck be removed?

The bottleneck is the directory accesses. With a central directory, all memory requests must be checked in a single location so as more processors are added, the directory has to be able to handle more and more requests. Directories can also be distributed across individual nodes. This way, directory requests can be directed at the node responsible for tracking the block rather than the central location.

e) In snooping protocols, what is the difference between invalidate schemes and update schemes? For each scheme, discuss a situation where the scheme performs poorly and explain why.

In invalidate schemes, when a processor modifies a block, it puts a message on the bus to inform all other caches that their version of that block is not current, thus telling them to move to the invalid state. In update schemes, on a write, the new value is put on the bus so all other caches which have a copy of the block can update their copies.
Invalidate schemes perform poorly when one processor is constantly modifying the value of a block while all other processors are reading the data. This is bad because each individual processor has to keep making requests for the data where in a update scheme a single broadcast would put the most recent value in all other caches.
Update schemes perform poorly when there are multiple writes to a single block which other caches may only need once in a while or may not need at all. This is bad because each write is broadcasted on the bus, where in an invalidate protocol, all other caches would be invalidated on the first write and no other communication would be needed.

2) (20 points)
(Textbook 4th edition Exercise 4.17, p.278) Directory protocols are more scalable than snooping protocols because they send explicit request and invalidate messages to those nodes that have copies of a block, while snooping protocols broadcast all requests and invalidates to all nodes. Consider the 16-processor system illustrated in Figure 4.42 and assume that all caches not shown have invalid blocks. For each of the sequences below, identify which nodes receive each request and invalidate.

a. P0: write 110 < -- 80
b. P0: write 108 < -- 88
c. P0: write 118 < -- 90
d. P0: write 128 < -- 98

a. No messages, hits in P0's cache
b. Send invalidate to P15
c. Send invalidate to P1
d. Send fetch/invalidate to P1

3) (20 points)
Consider the following hard drive:
Seek Time - 8 ms
Rotation Speed -7200 rpm
Transfer Rate - 80 MB/s
Controller Overhead - 0.1 ms
Sector Size – 512 bytes

a. We have a 1 MB binary file stored on a single track. The file contains sorted key/value pairs. Each pair consists of two integers. The first integer is the key and the second is its corresponding value. Integers are 4 bytes long. Assuming the integer key we are searching for is contained within the file, calculate the average time needed to find its corresponding value using sequential search. Assume we must do the search on disk, that is, we can only keep one sector (512 bytes) of the file in memory at any one time (e.g. we read 1 sector, process it, then read the next sector). Also, assume the only significant factor in the search time is the time to transfer sectors from disk (once transfered to memory, the time needed for to process each record is negligible).

On average, we'll have to sequentially read through half the file before we find the key/value pair.
This means well have to read 512 KB. Since we read by sectors and each sector is 512 bytes, we'll have to read 1024 sectors. Each read to a sector costs
Totaltime = seektime + rotationdelay + transfertime + overheadtime
= 8ms + (60000ms/1min) × (1min/7200rotations) ×.5rotation + (1000ms/80MB) × 512 bytes + 0.1ms
= 8ms + 4.17ms + 0.0061ms + 0.1ms = 12.27ms
With 1024 sector reads, that is 12570.73 ms total.

b. Consider the same setup as part a, except we can process the file as we read (instead of reading 1 sector, processing it, then going on to the next). On average, how long does it take to find a key/value pair on the hard disk?

Totaltime = seektime + rotationdelay + transfertime + overheadtime
= 8ms + (60000ms/1min) × (1min/7200rotations) ×.5rotation+ (1000ms/80MB) × 0.5MB + 0.1ms = 8ms + 4.17ms + 6.25ms + 0.1ms = 18.52ms

c. What would be the worst case time needed if we used binary search?

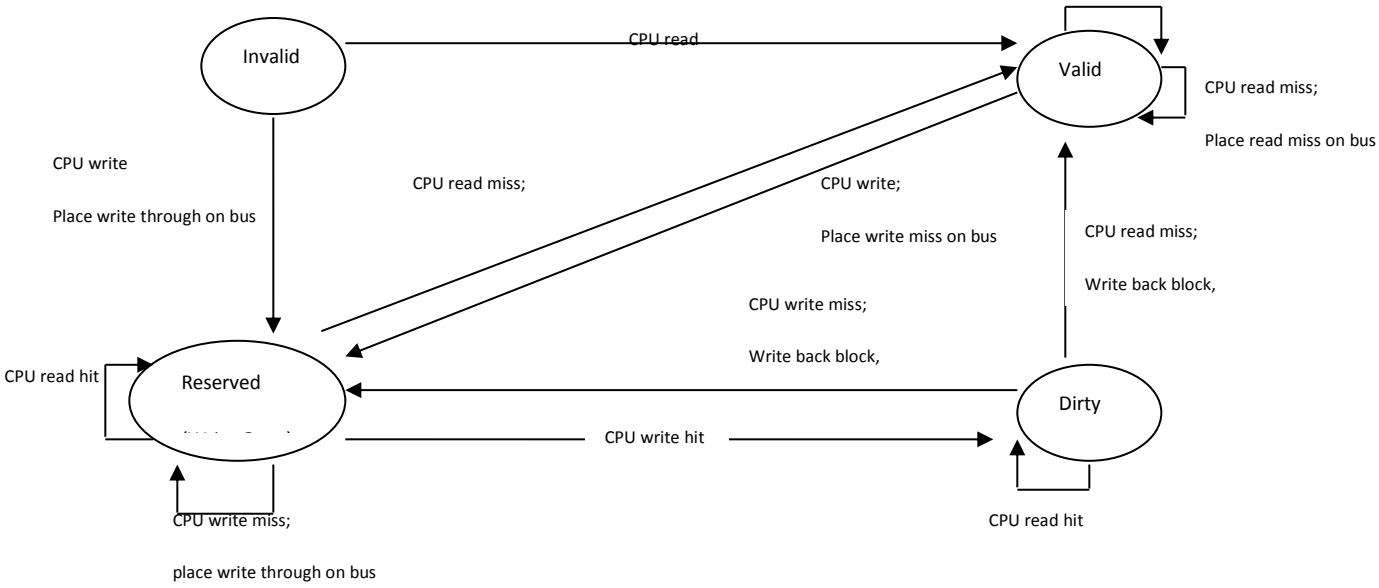
The file holds 1MB (1024 × 1024) / 8 = 131072 key/value pairs. If we used the binary search, it would take on average, $\log_2 131072 = 17$ accesses. However, the last 64 key/value pairs are all within the same 512 byte sector. So the last $\log_2(64) = 6$ accesses are all within the same sector, so effectively only one disk read is necessary for these last 6 accesses. That means there are a total of 12 reads for a total of 12.27ms × 12 = 147.24ms.
(Full credit given if you've accurately identified the number of reads required)

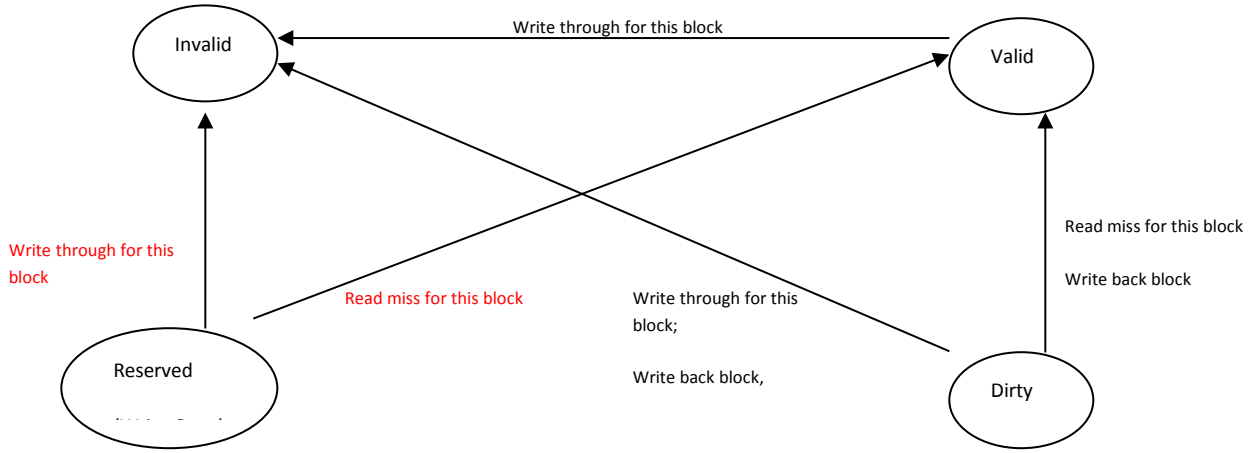
4) (40 points)

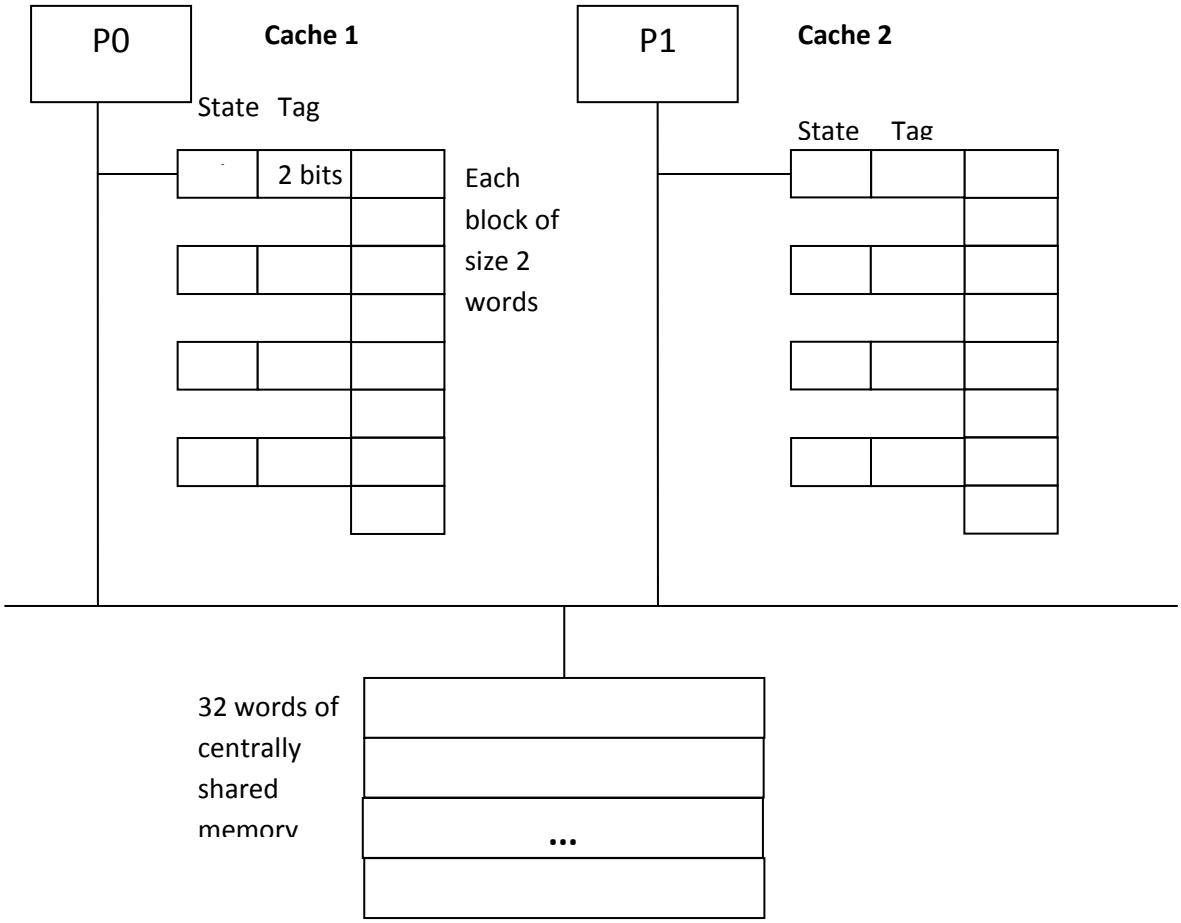
Consider the SNOOPING cache coherence protocol as follows.

Direct mapping is used. This scheme combines the advantages of both write-through and write-back invalidation. Only the very first write of a cache block uses a write-through policy. If a cache block is written more than once, it applies a write-back policy. There are four states in this protocol:

- **Invalid (Stage 00):** The block is not found in the cache.
- **Valid (Stage 01):** The cache block, which is consistent with the memory copy, has been read from shared memory and has not been modified.
- **Reserved (Stage 10):** Data has been written exactly once since it has been read from shared memory. The cache copy is consistent with the memory copy, which is the only other copy. (Write-through policy applies.)
- **Dirty (Stage 11):** The cache block has been modified (written) more than once, and the cache copy is the only one in the system (thus inconsistent with all other copies). (Write-back policy applies).



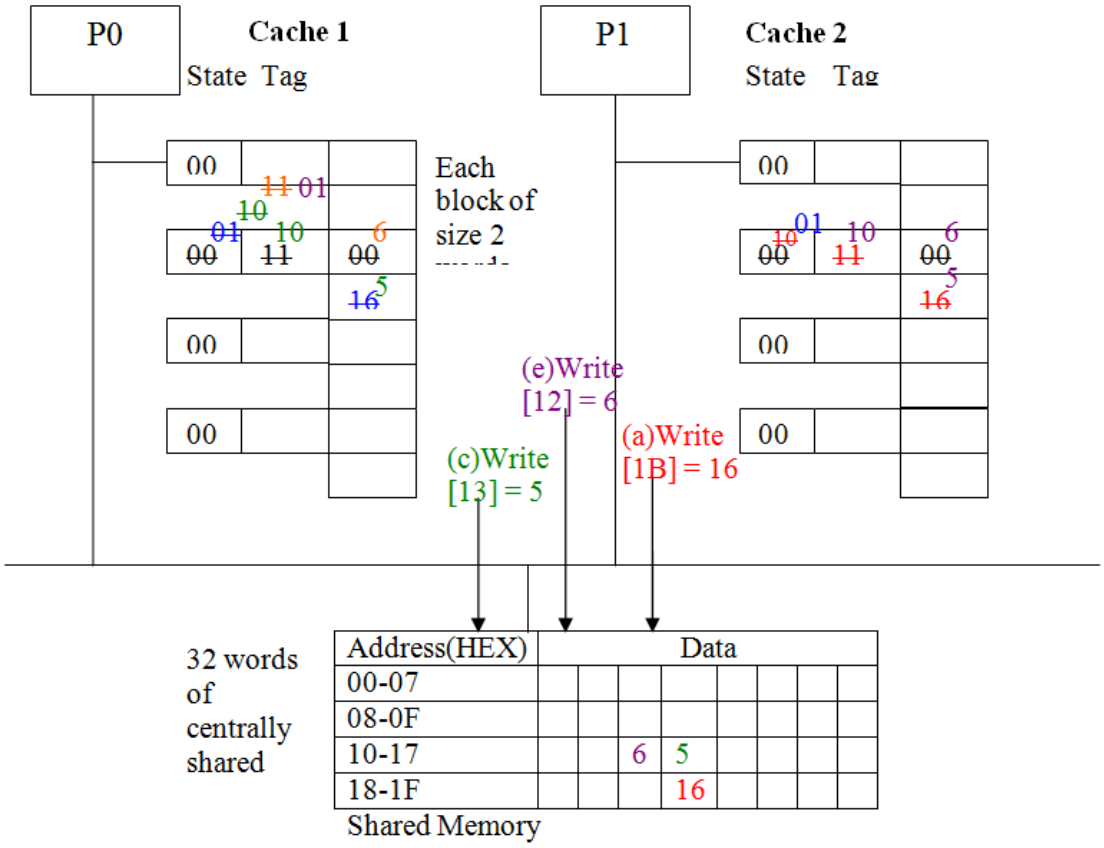




I. Assume that a multiprocessor has 2 nodes, P0 and P1. There is a 32-word centrally shared memory module. Each node has an 8-word local cache. Each memory block and each cache block have **2 words**. Draw the multiprocessor architecture. You have to specify the size of each block and the length of the tag field.

II. Trace the following events. Show all of the changes of the local cache blocks and the shared memory. You can directly mark the changes in the figure you have drawn in part 2). Please indicate the event identifier along with each change. Also, please determine if each of the events is a hit or miss. Note that Mem[x] refers to the access of the centrally shared memory, where x is the hexadecimal number of the WORD address. Initially, Mem[x]=0 for x can be any number between 00 and 1F. All of the local caches are in the invalid state initially.

Event id	Proc Id	Read/ Write	Word address (hexadecimal)	Value (decimal)	**Hit/miss? (you have to fill in your answers here)**
a	P1	Write	1B	16	MISS
b	P0	Read	1B		MISS
c	P0	Write	13	5	MISS
d	P0	Write	12	6	HIT
e	P1	Read	12		MISS



The colors represent different events as shown in the table of events above.