CS-537: Midterm Exam II: Midterm Harder (Spring 2001)

Please Read All Questions Carefully!

There are ten (10) total numbered pages

Name: ______________________________

Grading Page

	Points	Total Possible
1		25
2		25
3		30
4		20
Total		100

Name: ______________________________

Questions

All questions are in the long answer style. There are four total questions, and then you are free!

1. **MEMS-based Storage Scheduling. (25 points)**

   A new storage technology has been developed known as MEMS, short for micro-electromechanical stores. These tiny devices can be built to store much more data than traditional hard drives at a fraction of the cost, thanks to the wonders of nanotechnology. In this and other problems, we will assume that this style of MEMS device has replaced a traditional hard drive as our storage medium.

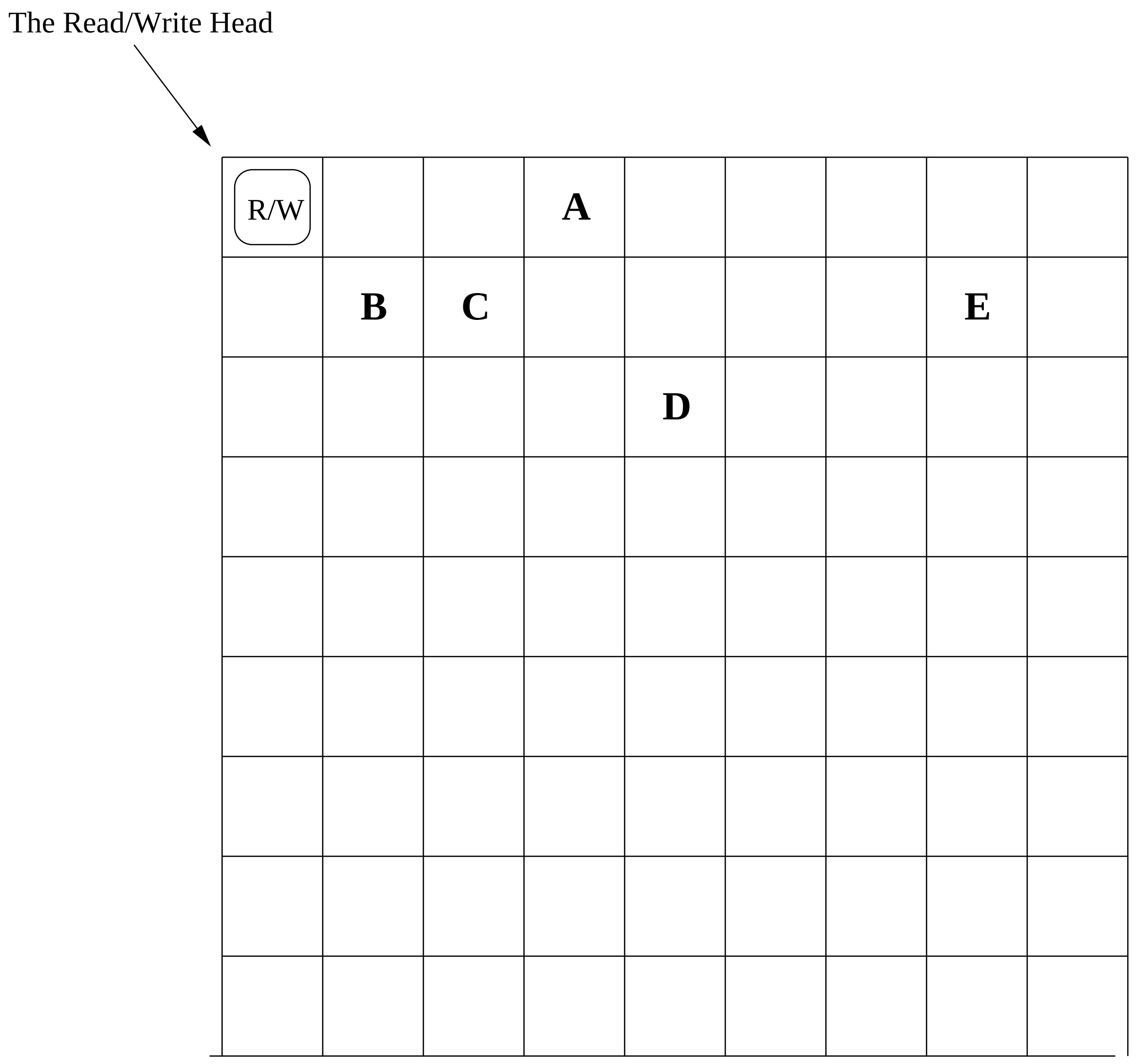
   This is how the MEMS-based storage technology will work: think of the device as a square grid, with each unit of the grid a block of data. A read/write unit floats above the grid (using magnetics), and can be moved to different spots in order to read or write a particular block. The read/write unit can only move horizontally or vertically (*i.e.*, it cannot move diagonally). Thus, in the picture on the next page, the read/write unit would have to move **three** units to get to block **C**: two right and one down. A typical MEMS might have 1024 rows and 1024 columns or so (roughly 1 million blocks) and each block might be 1 KB in size (1 GB total data).

   This first problem deals with MEMS scheduling.

   a) Assume the read/write unit starts at its current location in the upper-left corner of our 9 by 9 grid. Assuming a first-come, first-served discipline (FCFS), how long will it take to read all of the blocks, assuming that moving the read/write head a single unit in any direction takes 1 millisecond?

   b) Now, assume a shortest-access time first discipline (SATF). What order should you read A, B, C, D, and E now, assuming they all arrived at the same time, and how long will it take? Again assume the read/write head starts in the upper-left hand corner.

   c) Assuming SATF, what is the most worrisome problem that can occur under a steady stream of requests?



A 9x9 MEMS Grid

d) Now assume that moving the read/write unit vertically takes 3 times as long as to move it horizontally. Thus, each vertical move takes 3 milliseconds, whereas horizontal moves take only 1 millisecond. Now, what is the SATF ordering for A, B, C, D, and E, again assuming the read/write head starts in the upper left?

e) Assume you wish to implement a SCAN algorithm for the MEMS device. Describe the basic idea behind SCAN for disks, and then describe how you would apply it to the MEMS device. Again assume that the vertical move is 3 times as costly as the horizontal move.

2. **MEMS-FS: A File System For MEMS-based Storage. (25 points)**

   Now that you understand how to build a low-level disk scheduler for a MEMS device, you are put in charge of designing a file system for it. Fortunately, you are quite familiar with both the Berkeley Fast File System (FFS), and the Log-Structured File System (LFS), and maybe can apply those techniques here.

   a) FFS used the concept of a *cylinder group* to group together blocks that are related in some way, and thus likely to be accessed together. Describe how a cylinder group partitions a disk, and give a few examples of what FFS tries to put into a single cylinder group.

   b) Assuming that movement in either the vertical or horizontal direction is equally costly (say 1 millisecond per unit), how would you apply the cylinder group concept to your MEMS-based file system? Use a picture if you need to.

   c) Now let's think about applying LFS-like ideas to MEMS. If you want to organize the MEMS device as a log, how might you do so?

d) Traditional LFS groups many small writes into a single segment, and then writes that segment out to disk sequentially in order to obtain good performance from the disk. Would grouping many small writes into a single big segment be useful for the MEMS device? (explain) If so, why, and how big should a segment be? If not, why not?

e) Finally, you decide to use traditional file i-nodes and directories on the MEMS. Assuming you need to read the first block of the file `/home/remzi/537/grades.txt`, how many block reads will it take from the MEMS device? Assume that each directory's data fits into a single block, that each i-node is on a separate block, and that no blocks of any kind (i-node, normal file, directory file) are cached in main memory. Assuming the best possible layout on the MEMS device, and a 1 millisecond horizontal and vertical move cost, how long will this take?

3. **MEMS Virtual Memory (30 points).**

Now that you are a fully fledged MEMS hacker, you are asked to build the VM system for MEMS-based system too (yes, this is getting even worse than your 537 projects!). In this system, we have not only MEMS-based storage, but also very normal DRAM-based memory.

a) A common VM page replacement policy is least-recently-used, or LRU. Describe LRU and the intuition behind using it.

b) Unfortunately, LRU sometimes works very poorly. Describe a worst-case scenario for an LRU policy (you can show an example).

c) What replacement policy would work well in your worst-case LRU scenario?

d and e) In class, we often said that a *hybrid approach* can be used when both extremes do not work well, so maybe a hybrid policy could be developed here. Describe how such a hybrid policy could work. What are the key mechanisms? When should we switch between LRU and our other policy? Please be as detailed as possible.

f) Finally, assume that each block in your MEMS-device is 1 KB in size, and it takes 100 microseconds to access a single block (once the read/write unit is above that block in the grid). Also, assume the time to load or store a word (4 bytes) of data to our DRAM-based main memory is 1 microsecond. In this case, which is more expensive, the time to read 1 KB from the MEMS device, or the time to read 1 KB from main memory? From a performance standpoint, is it still worth having main memory in the system?

4. **Encrypted files on MEMS (20 points).**

   Finally, we also want to encrypt data in our MEMS-based file system, so that no one can read our precious files without the proper credentials. Assume we are using a *public key* system, and that the public key and secret key of user A are represented by the A_{public} and A_{secret}. Thus, a file encrypted in the secret key of A would be denoted $\{File\}_{A_{secret}}$.

   a) Assume user A wants to encrypt a particular file so that only user B can read it. Using the notation above, how would that file be encrypted?

   b) Now assume that user A wants to add a secret note (which we can call $Note$) for B to the file, and to ensure B that A is the user that encrypted the file. Also, user A doesn't want anyone but B to know what A has been up to. How would that be achieved?

   c and d) User C wants to create a file that only users A and B can read, *without* having to make a copy of the entire (large) file for each of them; we refer to this file as $file_{special}$. Assume that C can manufacture a new public/secret key pair if need be. Describe a scheme where C can achieve this goal (you can use more than one file if you want to, but you can only have a single copy of $file_{special}$).