CS-537: Final Exam (Fall 2011)
The Back of the Envelope

Please Read All Questions Carefully!

There are thirteen (13) total numbered pages.

Please put your NAME (mandatory) and student ID (optional) on THIS page only.

Name and Student ID: _____

> "Martin picked up from his desk a proposal for the communication system that his organization was building for the Summer Olympic games, and went through a sequence of basic calculations. He estimated one key parameter by measuring the time required to send himself a one-character piece of mail. The rest of his numbers were straight from the proposal and therefore quite precise. His calculations were simple but revealing. They showed that, under generous assumptions, the proposed system could work only if there were at least a hundred and twenty seconds in each minute. He sent the design back to the drawing board. This was Martin's wonderful (if eccentric) way of introducing me to back-of-the-envelope calculations." *Paraphrased from "The Back of the Envelope" by Jon Bentley*

Estimating how long something takes on the "back of the envelope" is a tried and true engineering technique. The idea is to get a rough estimate of something via paper and pencil, before having to build or analyze a full system. In this exam, you'll use your paper, pencil, and yes, brain, to do some of these types of calculations, to estimate how long certain operations take in a modern computer system. Read each question carefully, and good luck!

Grading Page

	Points	Total Possible
Q1		10
Q2		10
Q3		10
Q4		10
Q5		10
Q6		10
Q7		10
Q8		10
Q9		10
Q10		10
Total		100

1. **Virtual Memory.**

   In this question, we'll see how long a simple load instruction takes to execute, given some different assumptions about the virtual memory system.

   - Assume we have a linear page table per process, but the system we are using has no TLB and no swapping to disk. Assume each memory access takes M units of time. How long does it take to execute a single load instruction (e.g., `mov VirtualAddress, register`), in terms of memory accesses?

   - Now assume the same system, but with a TLB (still no swapping). Assume memory access is the dominant cost, and it takes M units of time to access memory. What is the fastest (i.e., best case) the load instruction above will execute, assuming TLB hits?

   - Assume now we have a two-level page table with a page directory. Assume each reference to the TLB is a miss, but that all referenced pages are found in memory (no swapping again). What is the slowest (i.e., worst case) time for the load instruction above?

   - Assume the same system again, but this time the memory location(s) referenced by the load instruction also might be on disk (due to swapping). Disk accesses (in this simple model) take D time units. What is the worst case time for the instruction to execute?

2. **Concurrency.**

   Given a basic spin lock, assume that locking the spin lock takes A time units (if no one is holding the lock); unlock also takes A time units. Assume further that a context switch takes C time units, and that a time slice is T time units long.

   Assume this code sequence, executed by two threads on one processor at roughly the same time:

```
mutex_lock();
do_something(); // takes no time to execute
mutex_unlock();
exit();         // takes no time to execute
```

   - What is the best-case time for the two threads on one CPU to finish this code sequence?

   - What is the worst-case time for the two threads to finish this code sequence? Assume that only three context switches can occur at a maximum.

   - If the spin lock is instead changed to a queue-based lock, how does that change the worst-case time?

3. **Disks.**

In this question, we'll perform some simple calculations on a highly-simplified disk.

- Assume a simple disk that has only a single track, and a simple FIFO scheduling policy. The rotational delay on this disk is R; there is no seek cost (only one track!), and transfer time is so fast we just consider it to be free. What is the (approximate) worst case execution time for three (3) requests (to different blocks)?

- Now assume that a shortest-access-time-first (SATF) scheduler is being used by the disk (but it still only has a single track). What is the worst-case time for three requests (to different blocks) now?

- Now assume the disk has three tracks. The time to seek between two adjacent tracks is S; it takes twice that to seek across two tracks (e.g., from the outer to the inner track). Given a FIFO scheduler, what is the worst-case time for three requests?

- Same question, but for a SATF scheduler.

4. **RAIDs.**

For this question, we'll examine how long it takes to perform a small workload consisting of 12 writes to random locations within a RAID. Assume that these random writes are spread "evenly" across the disks of the RAID. To begin with, assume a simple disk model where each read or write takes D time units.

- Assume we have a 4-disk RAID-0 (striping). How long does it take to complete the 12 writes?

- How long on a 4-disk RAID-1 (mirroring)?

- How long on a 4-disk RAID-4 (parity)?

- How long on a 4-disk RAID-5 (rotated parity)?

- Now assume we have a better disk model, in which it takes S time units to perform a random seek and R units of time to perform a full rotation; assume transfer is free. How long do the 12 random writes take to complete on a 4-disk RAID-0?

- How long on a 4-disk RAID-1 (mirroring)?

- How long on a 4-disk RAID-4 (parity)?

- How long on a 4-disk RAID-5 (rotated parity)?

5. **A Basic File System.**

For this question about basic file systems, assume a simple disk model where each disk read of a block takes D time units. Also assume the basic layout is quite like the very simple file system or FFS.

- Assume that all data and metadata begin on disk. Assume further that all inodes are in separate blocks, and that each directory is only one block in size. How long does it take to open the file `/a/b/c/d.txt`?

- Assume after opening the file, we read the file in its entirety. It is a big file, containing 1036 blocks. The inode itself has room for 12 direct pointers and 1 indirect pointer. Disk addresses are 4 bytes long, and disk blocks are 4KB in size. After opening it, how long does it take to read the entire file?

- Now assume a new inode structure is introduced, in which there is only one pointer: a double indirect pointer, which points to the double indirect block, which can point to 1024 indirect blocks, each of which can point to 1024 blocks. After opening the file, how long does it take to perform 50 random reads within a very large file?

- How long does it take to close a file, approximately (assuming disk accesses are the dominant cost)?

6. **Journaling File Systems.**

For this question about journaling file systems, assume the following disk model: it takes S time units to seek to any location on the disk; the full rotational delay is R time units; transferring data takes T units, regardless of the amount of data transferred.

- Assume we have a data journaling file system in which all data and metadata are logged before being written to disk. How long does it take to append a block to an existing file, assuming that the all relevant metadata and data was in the OS page cache to begin with (i.e., no reads from disk need occur)?

- Now assume we have journaling, but only for metadata; data blocks are written directly to their final location. How long does the append take in this case?

- Now do the same calculation, but without journaling at all. How long does the append take without journaling?

- Finally, let's assume we're using full data journaling again. How long does it take to create a new empty file? Again assume that all relevant structures are in cache to begin with.

7. **LFS, the Log-structured File System.**

We'll now analyze the performance of LFS, the log-structured file system. Assume it takes S time units to seek to a segment, and T time units to read or write it after that seek. If, however, we access segments in sequential order (segment 0, then 1, then 2, ...), no seek cost is incurred.

- Assume we have a newly-created empty file system. We now write out a file that fills 100 segments. How long does it take to complete this write sequence?

- Assume we now read this file (and that the reads do not hit in the memory cache); how long does it take to read the entire file?

- Assume we now read the file backwards, one segment at a time (and that the reads do not hit in the memory cache); how long does this backwards read take?

- Assume we have an old file system, in which many segments have been repeatedly cleaned, leaving a few hundred free segments scattered over the disk. How long does it take to write out 100 segments on this file system?

- Assume now that the disk is nearly full, and that to write a segment, we have to clean two segments (which produces one compact segment, and one free one). In this case, how long does it take to write out 100 segments?

8. **NFS, Sun's Network File System.**

We'll now model the time of certain operations in the Sun Network File System. The only costs to worry about are network costs; assume everything else is free. Assume any "small" message takes S units of time to be sent from one machine to another, whereas a "bigger" message (e.g., the size of a disk block, 4KB) takes B time units. If a message is larger than 4KB (e.g., 8KB), it should take proportionally longer (e.g., $2B$ for 8KB).

- How long does it take to open a 100-block (400 KB) file called `/a/b/c.txt` for the first time? (assume that the root directory is indeed the root of the NFS file system)

- How long does it take to read the file?

- How long does it take to re-read that file immediately?

- How long does it take to re-read that file after a little while?

- How long does it take to re-read that file a long time later, after many other memory-consuming programs have run on the client?

9. **AFS, the Andrew File System.**

We'll now model the time of certain operations in the Andrew File System. The only costs to worry about are network costs; assume everything else is free. Assume any "small" message takes S units of time to be sent from one machine to another, whereas a "bigger" message (e.g., the size of a disk block, 4KB) takes B time units. If a message is larger than 4KB (e.g., 8KB), it should take proportionally longer (e.g., $2B$ for 8KB).

- How long does it take to open a 100-block (400 KB) file called `/a/b/c.txt` for the first time? (assume that the root directory is indeed the root of the AFS file system)

- How long does it take to read the file?

- How long does it take to re-read that file immediately?

- How long does it take to re-read that file after a little while?

- How long does it take to re-read that file a long time later, after many other memory-consuming programs have run on the client?

10. **Virtual Machine Monitors.**

With virtual machine monitors (VMMs), a lot of excess overhead arises due to the extra level of virtualization beneath the OS. In this question, we'll examine these overheads. Assume the only costs we are concerned with are T, the cost of trapping into the VMM, and R, the cost of returning from such a trap.

- When performing a system call on an OS *not* running on a VMM, how long does it take in terms of T and R?

- When performing a system call on an OS running on a VMM, how long does it take in terms of T and R? (assume the system call is a simple one, such as getpid())

- When executing an instruction that causes a TLB miss (assuming a software-managed TLB), how long does it take on an OS *not* running on a VMM?

- When executing an instruction that causes a TLB miss (assuming a software-managed TLB), how long does it take on an OS running on a VMM?