CSE 451 Midterm Exam #2
February 16th, 2001

Your Name: _____

Student ID: _____

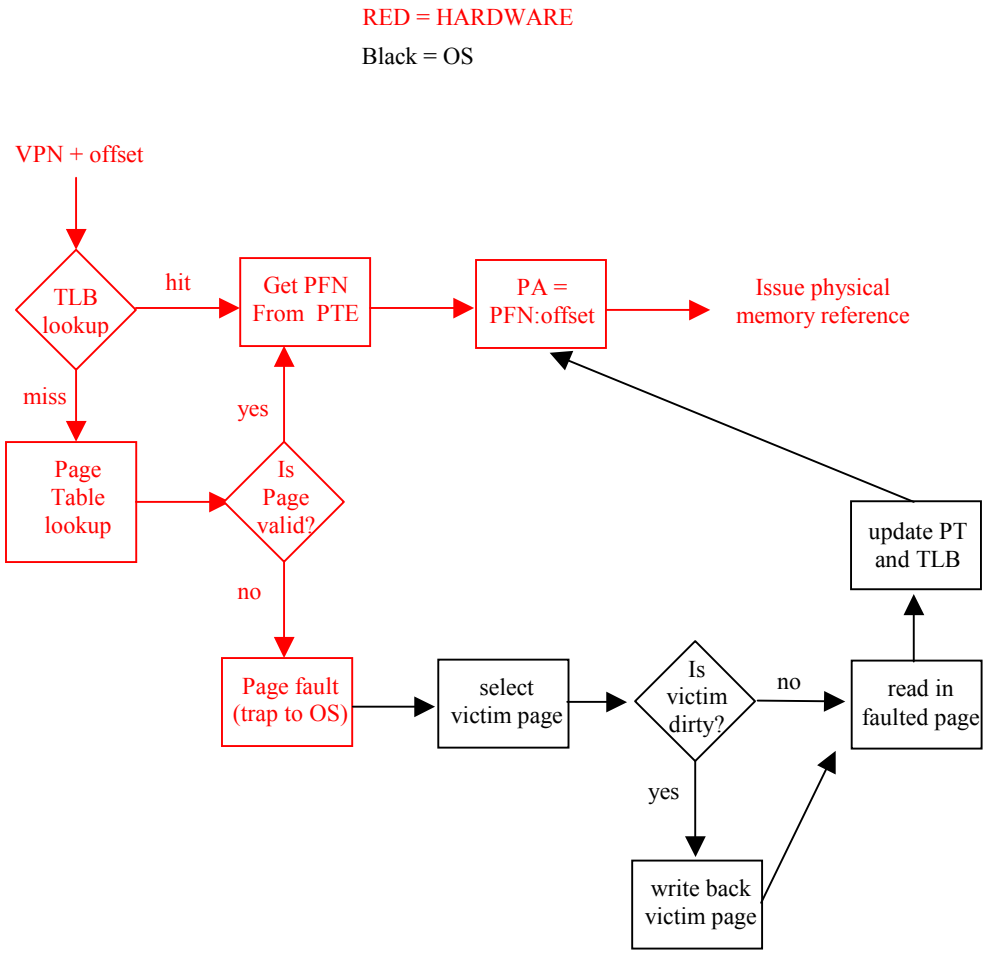
General Information:

This is a closed book examination. You have **50 minutes** to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points given to the question. There are **6 pages** on this exam (check to make sure you have all of them), and there is a total of **100 points** in all. Write all of your answers directly on this paper. Make your answers as concise as possible. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer.

Problem 1: (25 points)

Consider an operating system that uses paging in order to provide virtual memory capability; the paging system employs both a TLB and a single-level page table. Assume that page tables are "wired" (pinned) in physical memory.

Draw a flow chart that describes the logic of handling a memory reference. Your chart must include the possibility of TLB hits and misses, as well as page faults. Be sure to mark which activities are accomplished by hardware, and which are accomplished by the OS's page fault exception handler.



Problem 2: (20 points)

Supposed we have an OS that uses paged virtual memory, and further suppose that the OS stores the page table in CPU registers. (Thus, in this problem, you may assume that looking into the page table has negligible time cost.) Assume that it takes 8 milliseconds to service a page fault if an empty page frame is available or if the victim page is not dirty, and 20 milliseconds if the victim page is dirty. Suppose further that physical memory takes 1 microsecond to access.

If the page replacement algorithm selects a dirty victim page 60 percent of the time, then what is the maximum acceptable page fault rate for an effective access time of no more than 2 microseconds? Express the page fault rate "p" as the probability that a memory reference results in a page fault.

Leave your answer in the form of an arithmetic expression, e.g.:

max-acceptable-page-fault-rate = (12345 – 3.3)/4.4 + 3*35

average page fault time = (probability of a fault) * (fault service time) +
(probability of a non-fault) * (memory access time)

= p * [0.4 * 8ms + 0.6 * 20ms] + (1-p)*0.001ms
= p * [15.199ms] + .001ms

We want average page fault time to be 2 microseconds = .002 ms

0.002ms = p*[15.199ms] + 0.001ms

p = 0.001ms / 15.199ms

Problem 3: (25 points)

i) A friend of yours knows that you are an expert on file systems, and so he comes to you with the following observation. He tells you that while he was using SteveFS (the latest, greatest file system implementation), he observed that in general, file reads tended to proceed faster than file writes. Propose three hypotheses for why this might be true:

1. The buffer cache is write-through, thus all writes are synchronous.
2. Writes are small (e.g. 1 byte), forcing the FS to read a block, modify the one byte, then write the entire block back. In this case, there are two block accesses per write.
3. Writes require modifying the free list, and this is expensive for SteveFS.
4. Writes require modifying inodes, which is an extra write operation per file modification.
5. Appending to a file forces SteveFS to find a free block, which happens to be expensive (i.e., the free list is inefficient to access).
6. The FS implementation uses sequential block allocation, so creating a new file on a write requires an expensive search to find a "hole" of the right size.
7. The friend's workload happens to be such that reads have high locality (thus high hit rates in the buffer cache), but writes have poor locality (thus low hit rates in the buffer cache).
8. SteveFS prefetches data on reads, making them faster on average.
9. SteveFS enforces file consistency, and thus if two programs write to the same file ("write sharing"), one is blocked while the other's write proceeds.
10. SteveFS moves files that are frequently read to the middle cylinders of the disk, minimizing average seek time to get to them.
11. The disk is nearly full. Block allocation on writes may be more expensive, and new files being written will be fragmented across the disk. Existing files (which are read) however will be mostly sequential.

ii) A different friend of yours also knows that you are an expert on file systems, and so she comes to you with the following observation. She tells you that while she was using GribbleFS (a competitor to SteveFS), she observed that in general, file writes tended to proceed faster than file reads. Propose two hypotheses for why this might be true:

1. GribbleFS uses a delayed-write buffer cache, and files tend to be overwritten. Thus, write traffic is often absorbed by the buffer cache instead of actually making it out to disk.
2. GribbleFS prioritizes handling writes over handling reads; thus, if both a read and a write are pending, the read is delayed until the write succeeds.
3. GribbleFS places writes in the nearest free block to wherever the head happens to be at the time of the write. This means writes are fast (small seeks), but reads will involve large seeks.
4. GribbleFS is really LFS, but rebranded. ;)
5. GribbleFS has no read cache.

Problem 4: (30 points)

Consider a disk with the following physical characteristics:

capacity = 36GB = 36 * 2^{30} bytes

 # cylinders = 12,000; assume that all cylinders have the same capacity

worst-case seek time = 10 milliseconds; assume seek time is proportional to the number of cylinders that are spanned during the seek

sequential read/write transfer rate = 15 MB/s = 15 * 2^{20} bytes per second

rotational speed = 7200 revolutions per minute = 120 revolutions per second

disk block size = 4KB = 4 * 2^{10} bytes

i) **(4 points)** Assuming that traffic is perfectly random, what is the average seek time of this device? (Hint: you answered this on homework #3. If you don't know the answer, don't spend time on this part of the question.)

Average seek length = 1/3 worst-case seek length = 4000 cylinders
Average seek time = (4000/12000) * worst case seek time = 10/3 ms

ii) **(6 points)** What is the average rotational latency of this device?

Rotational speed = 120 revolutions per second, therefore 1 rotation takes 1/120 s

Avg rotational latency = duration of ½ rotation = 1/240 s = 25/6 ms = 4.16 ms

(question continues on the next page)

iii) **(10 points)** Assume that the disk blocks for a given file are randomly scattered across the disk. On average, how long does it take to read a 5 MB file? Express your answer symbolically (i.e., in terms of average seek time, etc.) as well as numerically.

A 5MB file contains 5MB/4KB blocks = $5*2^{20} / 4*2^{10} = (5/4) * 2^{10}$ blocks
= 1280 blocks

Once the head is in position, the time it takes to read 1 block = read length divided by sequential read bandwidth
= 4KB / 15 MB/s = $(4/15) * 2^{-10}$ seconds

Since blocks are randomly scattered, to position the head for the a block, you will suffer the average seek time + the average rotational latency

= 10/3 ms + 25/6 ms = 15/2 ms = 15/2000 s = 3/400 s

Total time to read a file = # blocks * time to read a block
= [1280] * [3/400 + $(4/15)*2^{-10}$]s
= 9.933333 s

iv) **(10 points)** Assume that the disk blocks for a given file are sequentially placed on disk, i.e. that the first block of the file is located on a randomly allocated cylinder, and that successive blocks of the file are allocated sequentially from the same cylinder, or from successive cylinders (if the file is larger than a single cylinder).

On average, how long does it take to read a 5 MB file? Express your answer symbolically as well as numerically.

A 36GB disk with 12000 cylinders has:
$36*2^{30} / 12000$ = 3221225.472 bytes per cylinder

Thus, a 5MB file will fill one cylinder, and have (5MB-3221225.472) =
= 2021654.528 bytes spilling over onto the next cylinder

To seek the head to the initial cylinder holding the file takes 10/3 ms.

Then, you must wait average rotational latency = 4.16 ms before first block of file rotates to under the head.

Given a 15MB/s sequential throughput, it then takes (3221225.472/15*1024*1024)s = 204.8ms to read first cylinder's worth of file.

Afterwards, you must seek one cylinder, which takes 1/1200 ms.

Then, you must wait average rotational latency = 4.16 ms for first block of file (on second cylinder) to spin under head.

Finally, you must wait (2021654.528/15*1024*1024)s = 128.53ms to read the remainder of the file (spin under head).

So, total latency is:

= (10/3ms) + 4.16ms + 204.8ms + (1/1200ms) + 4.16ms + 128.53ms

= 344.984ms