1. True/False Problems

Answer the following true/false (T/F) questions. Each T/F problem is worth 2 points. Each missing or wrong answer costs -2 point.

(a) The dispatcher is responsible for setting thread priorities.
(b) If a thread is CPU-bound, it makes sense to give it higher priority for disk I/O than an I/Obound thread.
(c) Virtual addresses must be same size as physical addresses
(d) Page offsets in virtual addresses must be the same size as page offsets in physical addresses.
(e) Implementing processor affinity in a multiprocessor scheduler is likely to reduce the number of misses in caches such as translation lookaside buffers.
(f) With demand paging, adding memory to a system always improves the cache hit rate.
(g) It is not possible to implement user-kernel separation without hardware support for dual mode operation.
(h) Without “free” (deallocate) it is easy to write a “malloc” implementation that never fragments memory.
(i) SSD reads always outperform reads to a spinning media hard drive
(j) SSD writes always outperform writes to a spinning media hard drive
(k) Writes to an SSD are significantly faster than Reads from an SSD.
(l) DMA allows an I/O device to transfer data to and from memory without involving the CPU in the transfer.
(m) Memory mapped I/O maps disk blocks to page frames.
(n) There is only one MBR (master boot record) on a disk drive, but there could be several boot sectors.
(o) A context switch from one process to another can be accomplished without executing OS code in kernel mode.
(p) On the x86 architecture, if a given memory reference (load or store) causes a TLB miss, then that memory reference also causes a page fault.
(q) The Elevator disk scheduling algorithm bounds the waiting time for all disk requests.
(r) An entry in the open file table maintains a pointer to an in-memory inode structure.
(s) When a process issues a system call, the OS code starts by executing an instruction to change the processor mode (from user to kernel).

2. Short Questions

(a) Suppose you run a workload using a fixed-size 1 MB (megabyte) buffer cache and notice that there are many disk references. If you repartition main memory to increase the buffer cache size to 2 MB, is it possible that the workload actually run significantly slower? Explain why. (2 points)

(b) Is address translation (virtual memory) useful even if the total size of virtual memory as summed over all possible running programs is guaranteed to be smaller than a

machine's total physical memory? Why? (2 points)

(c) If the block size is doubled in the UNIX file system, will this double the maximum file size? Explain why? (2 points)

(d) Explain the reasoning behind the use of the LRU algorithm for virtual memory page replacement and file system buffer caches. (2 points)

(e) You overhear one of your classmates saying that any space considered internal fragmentation must not be on the free list while any external fragmentation must be on the free list. Is this statement correct? Justify your answer. (2 points)

(f) What is a TLB and what does it do? (2 points)

(g) In class, we discussed copy-on-write for memory pages shared among multiple processes. Why is copy-on-write potentially better than copying the entire process immediately upon creation? (2 points)

(h) Which of the following operating systems use the optimal memory page replacement algorithm? Windows 7/10, Solaris (Sun UNIX), Linux, iOS, and Android. (2 points)

3. Choose the best answer for the following question. (2 points)

Counting semaphores:

(a) generalize the notion of a binary semaphore
(b) are used for managing multiple instances of a resource
(c) have increment and decrement operations
(d) can use queueing to manage waiting processes
(e) all of the above

4. Consider the following list of actions. Put a check mark in the blank beside those actions that should be performed by the kernel, and not by user programs. Put at most 4 marks. (6 points; any additional mark beyond 4 takes 1.5 point off each)
    - _____ reading the value of the program counter (PC).
    - _____ changing the value of the program counter (PC).
    - _____ changing the value of the segment table base register.
    - _____ changing the value of the stack pointer (SP).
    - _____ increasing the size of an address space.
    - _____ creating a memory segment that is shared between multiple processes.
    - _____ writing to a memory segment that is shared between multiple processes.
    - _____ disabling interrupts.

5. Processes (or threads) can be in one of three states: Running, Ready, or Blocked. In which state is the process (or thread) for each of the following four cases? (3 points)

    (i) Waiting for data to be read from a disk.
    (ii) Spin-waiting for a lock to be released.
    (iii) Having just completed an I/O and waiting to get scheduled again on the CPU.

6. Why is switching threads less costly than switching processes? (2 points)

7. Suppose a thread is running in a critical section of code, meaning that it has acquired all the locks through proper arbitration. Can it get context switched? Why or why not? (3 points)

8. Why would two processes want to use shared memory for communication instead of using message passing? (2 points)

9. (a) If a normal access to memory takes 100 ns (100×10^-9 sec) and reading a page from disk takes 10 ms (10×10^-3 sec), what is the average memory access time (including page fault overhead) if page faults occur in 0.1% of the memory references? You may assume that there is no additional overhead due to TLB misses. (2 points)

(b) What is the maximum allowable page fault rate if performance degradation is to be no more than 10% (i.e. average memory access time ≤ 110 ns)? (2 points)

10. Assuming a page size of 1 KB and that each page table entry (PTE) takes 4 bytes, how many levels of page tables would be required to map a 34-bit address if every page table fits into a single page. Be explicit in your explanation. (7 points)

11. The Google Android Operating System uses a Linux kernel to support its applications. Each application runs in its own process. Android devices do not typically have a disk, and they (typically) do not use swapping to stretch the amount of available physical memory when they run short. Instead, the OS may terminate processes, requiring the applications to store and restore their state when this happens. Under these circumstances, would Android still derive any benefits from Linux's virtual memory management scheme? Justify your answer! (4 points)