Operating Systems/Architecture Candidacy Exam SP 2024

Name: ____

PSU ID: ______________________________ **PSU email**: _________________@psu.edu

Instructions: This exam contains a collection of questions from two areas (*OS* and *Arch*). Each question is worth **20** points. Select below, by marking the appropriate box in the "**Skipped"** column, **one** OS and **one** Arch question to **NOT** be graded. You will have **3 hours** to answer the remaining **80** points worth of questions **from each area** for a total of **160 pts.** You **MUST** select a question to be skipped: Failure to indicate which question is to be skipped during grading will result in your highest scoring question being set to **0 points**.

Please provide answers in the space provided for that question (exams will be scanned into Gradescope); answers outside the provided space may not be graded.

The exam is closed book, so no notes, phones, etc. Good luck!

Questions			**Skipped?**
OS	**OSQ1:**	**Kernel (20 pts)**	
	OSQ2:	**Scheduling (20 pts)**	
	OSQ3:	**Concurrency and Synchronization (20 pts)**	
	OSQ4:	**Storage Systems (20 pts)**	
	OSQ5:	**Virtual Memory (20 pts)**	
Arch	**AQ1:**	**Multithreading and False Sharing (20 pts)**	
	AQ2:	**CPU Performance (20 pts)**	
	AQ3:	**Memory Accesses and Optimization (20 pts)**	
	AQ4:	**Architecture Reliability (20 pts)**	
	AQ5:	**Instruction Pipelining (20 pts)**	
Total = (4 * 20 pts) + (4 * 20 pts) = 160 pts possible			

Operating Systems

{OSQ1} {Kernel} {20 pts}

(a) **{5 pts}** The OS allows user programs to execute some code that resides in the kernel (e.g., open(), read(), mmap()).

i) Why can't these be ordinary function calls, like with an external library?

ii) What is the mechanism used to allow user programs to execute these code regions?.

(b) **{6 pts}** Explain the difference between processes and kernel-level threads. Additionally, select all of the below actions that **do** occur when context switching from thread A to thread B in a different process, but **do not** occur when switching between threads A and B in the same process.

☐ Thread A's registers are saved	☐ Thread B's stack is loaded into memory
☐ Thread A's registers are restored	☐ The base page table register is switched
☐ Thread B's registers are saved	☐ The TLB is flushed
☐ Thread B's registers are restored	☐ The kernel stack register is switched
☐ The user stacks of the threads are swapped in memory	☐ The stack register is switched

(c) **{4 pts}** Give two examples of an exception (trap), where

i) the first will almost always cause a user process to be terminated (and why)

ii) the second will almost never cause a user process to be terminated (and why)

(d) **{5 pts}** In typical OSs, when handling an interrupt or trap while the CPU is in user mode, the CPU switches to a dedicated kernel stack.

i) How does the CPU know where the kernel stack is? Pick **one** of the following choices.

Determined based on thread id	Always starts at a predefined location	Looked up in a kernel data structure
Specified in kernel stack register	Location is configured during bootup	

ii) Describe a security challenge that would occur if the CPU used the existing stack to execute a corresponding interrupt or trap handler.

{OSQ1 answer space, continued} {Kernel}

{OSQ2} {Scheduling} {20pts}

(a) **{4pts}** Consider we have four jobs with (arrival-time, runtime) pairs specified as (0, 1000), (100, 500), (400, 200), (0, 600). Compute the average response time and turnaround time for Shortest-Job-First, Shortest-Time-to-Completion-First, and Round-Robin with 200 time units as the scheduling quantum (time slice). Here response time means the time between when a job arrives and when it gets first scheduled, while turnaround time means the time between when job arrival time and when it finishes.

(b) **{4 pts}** Explain what happens when a parent process terminates before its children.

(c) **{4 pts}** Explain why a multi-level feedback queue scheduler can possibly cause starvation and provide a solution to this problem.

(d) **{4pts}** Answer the following prompts relating to I/O and preemption

i) define "polling" and when it is beneficial to use for I/O event discovery/interactions

ii) define "I/O interrupts" and when they are beneficial to use for I/O event discovery/interactions

iii) why is it problematic for the CPU and OS to always wait until an I/O request completes?

iv) do OSs have to support interrupts and preemption? If not, how would they work without it?

(e) **{4 pts}** Assume a set of 256 locks, each identified by a pair of coordinates (x, y) where x is in range [0,15] and y is in range [0,15]. Explain why deadlock (due to locking) will not occur if all participating threads acquire locks using 16*x + y as their lock acquisition ordering function.

{OSQ2 answer space, continued} {Scheduling}

{OSQ3} {Concurrency and Synchronization} {20pts}

A group of **K** (assume K >= 1) students is studying for the Arch/OS qualifying exam. None of these students are able to study unless they are eating pizza. Each pizza consists of **S** slices, and slices cannot be shared among multiple students. If a student attempts to pick up a slice of pizza, and there is no pizza on the table, they immediately collapse from exhaustion and sleep until a new pizza arrives. The student who takes the last slice of pizza must order another pizza before beginning to study. Each of the **K** students executes the following loop:

```
while (true) {
  pickUpSliceFromTable(); // picks up a slice of pizza from the table
  study (); // study; consumes slice of pizza
}
```

Write code to synchronize the student threads (multiple) and the pizza order/delivery thread (singular). Your solution must avoid deadlock and ensure that only one new pizza is ordered each time the current pizza is consumed.

{OSQ3 answer space, continued} {Concurrency and Synchronization}

{OSQ4} {Storage} {20 pts}

(a) **{6 pts}** Suppose all file system caches (e.g., block cache, inode cache, etc.) are cleared and all files in this file system only contain one data block, but that repeated accesses will hit in associated caches and will not cause an IO operation (To get partial credit, describe what each IO is for in addition to providing the total count).

> i) If "/foo/bar" is a symbolic (soft) link to "/baz/garply", how many IOs will be triggered during the system call `open("/foo/bar", O_RDONLY)`?
> ii) How would this answer differ if "/foo/bar" was a hardlink?

(c) **{6 pts}** For a file system with 4KB blocks and 8 byte block numbers, suppose each inode contains 6 direct block numbers, 1 single indirect block number, and 1 double indirect block number.

> i) How many block numbers can one single indirect block contain?
> ii) Consider a file in this file system containing 31MB of data. How many total indirect blocks (single and double) will be part of the file's metadata?

(d) **{4 pts}** Assume that, due to an abrupt power outage, your computer has crashed in the middle of performing file IO. Will your file system return to a consistent state faster if it uses i) an fsck-based or ii) a journaling approach, and why?

(e) **{4 pts}** Briefly describe how RAID-5 enables access to data after a single disk failure and what will happen if
> i) the new disk fails during restoration
> ii) one of the old disks fails during restoration.

{OSQ4 answer space, continued} {Storage}

{OSQ5} {Virtual Memory} {20 pts}

(a) **{6 pts}** Modern systems typically employ multi-level page tables, while single-level page tables were once common.

i) Assuming 8B page table entries, 16KiB page size, a 48 bit virtual address, and a 40 bit physical address, how much memory would a single-level page table occupy?

ii) Suppose a different system features a multi-level page table with an 8KB page size and 8-byte PTE size. How many layers should this page table have to support a 64-bit virtual address space?

iii) Based on the two answers above, list one benefit of multi-level page-tables over single level page tables and one drawback of multi-level page tables over single level page tables.

(b)**: {4 pts}** Each process typically has its own page table. Can a process update its own page table without involving the OS kernel, i.e., directly from user space? Please explain why "yes" or why "no".

(c) **{4 pts}** Aside from leveraging secondary storage (HDD, SSD, etc.) to increase the total effective memory size, there are other important benefits of virtual memory in modern operating systems. Describe at least one such benefit.

(d) **{6 pts}** Given the page mapping shown below (note, this is the logical representation of the relevant contents of multiple page tables, not an actual page table), and assuming 4KiB pages, with page (virtual) and frame (physical) specified in decimal:

Process #	**Page #**	**Frame #**	**Ref**	**Dirty**	**Permission**
2	8	11	1	0	r
1	5	7	1	1	rw
3	9	6	0	1	rw
1	11	18	0	1	rw
3	8	12	1	0	rx
2	2	7	1	1	rw
2	9	18	0	1	rw
1	20	12	1	0	rx

i) What physical memory address is accessed when process 1 accesses the virtual address **0xb321**?

ii) What other (process, virtual address) pair could access the same physical address for a process **not** equal to 1?

{OSQ5 answer space, continued} {Virtual Memory}

{AQ1} {Multithreading and False Sharing} {20pts}

In a multithreaded execution, “false sharing” occurs when different memory addresses that belong to different threads are assigned to the same cache line. In such cases, updating one of the cache lines causes another to be kicked out of the cache, resulting in additional cache misses.

(a) **{4pts}** Give a sample (multithreaded) code fragment illustrating a case in which false sharing may occur. Use, if needed, illustrations to explain how it may happen.
(b) **{4pts}** Suggest a software strategy (e.g., one that can be adopted by a compiler) to minimize false sharing.
(c) **{4pts}** Suggest a hardware strategy (e.g., setting certain hardware design parameters) to minimize false sharing. In both (b) and (c), explain the potential issues (costs) that come with false sharing minimization.
(d) **{4pts}** Explain the impact on false sharing of i) increasing the number of bytes per cache block (line), ii) increasing the number of cache sets; iii) increasing the number of cache ways; and iv) increasing the number of threads.
(e) **{4pts}** Explain whether false sharing is the *only* source of “coherence misses”.

{AQ1 answer space, continued} {Multithreading and False Sharing}

{AQ2} {CPU Performance} {20pts}

(a) **{10pts}** Suppose a processor with only one data cache and one instruction cache executes at 400MHz clock rate with ideal CPI = 1. Imagine a program that contains 50% arithmetic/logic instructions, 30% load/store operations, and 20% control transfer instructions (conditional/unconditional branches and procedure jumps). If the miss rate for data cache is 20% and that for instruction cache 5%, both with 200 cycles miss penalty, what is the average CPI of the program?

{AQ2, continued} {CPU Performance}

(b) **{10pts}** A program has two parallel segments and one sequential (serial) segment. The two parallel segments make up 40% and 20% of the program. When executing this program on a processor with 8 cores, we observe speedups of 4 and 5 for the two parallel segments, respectively. What is the “overall speedup” for the entire program?

{AQ3} {Memory Accesses and Optimization} {20pts}

The performance of a "hierarchical memory hierarchy" is primarily determined by the *effective memory access time* (called T_{eff}), which depends on hit ratios and access frequencies at different levels. Suppose we have a hierarchical data memory system with memories M_1, M_2, …, M_n and a data access strategy which accesses these memories from M_1 to M_n in-order until the requested data item is found. The hit ratio and access frequency to the memory at level i (M_i) are h_i and f_i, respectively.

(a) **{4pts}** Define access frequency f_i in terms of h_1 through h_i.
(b) **{4pts}** Considering the property of data locality, can you express the relationships among f_1, f_2, …, f_n.
(c) **{4pts}** Express T_{eff} of the entire memory hierarchy in terms of hit ratios (to different levels) and access frequencies.
(d) **{4pts}** Assuming c_i and s_i refer to the per-KB cost and capacity (in KBs) of M_i, how can you express the total cost (C_{tot}) of the memory hierarchy? What is the typical relationship among c_1, c_2, …, c_n and that among s_1, s_2, …, s_n.
(e) **{4pts}** The optimal design of a memory hierarchy should result in a T_{eff} close to the t_1 of M_1 and a total cost close to the c_n of M_n. Can you express a "memory hierarchy optimization problem", in terms of T_{eff} and C_{tot}?

{AQ3 answer space, continued} {Memory Accesses and Optimization}

{AQ4} {Architecture Reliability} {20pts}

One of the major issues facing computer architects and software designers alike is "hardware errors". Suppose you designed a new microprocessor, but unfortunately it came back from the fab with an error: *one of the bits is stuck*. Sometime a bit can get stuck at 1 and sometime at 0. For each of the hardware components below, explain whether a stuck-at-0 or a stuck-at-1 fault would affect the correctness of the chip (and eventually the execution) and how. You can assume that the architecture does *not* have an error mitigation strategy implemented.

(a) **{4pts}** A bit in one of adders in ALU.
(b) **{4pts}** LRU bit(s) for one of the sets of a 4-way set-associate L2 cache.
(c) **{4pts}** A bit in branch prediction hardware (consider all possibilities).
(d) **{4pts}** A dirty bit of an L3 cache block (line).
(e) **{4pts}** A bit in one of the pipeline registers (consider all possibilities).

{AQ4 answer space, continued} {Architecture Reliability}

{AQ5} {Instruction Pipelining} {20pts}

A hardware vendor has designed a new 10-stage instruction pipeline:

$$IF \rightarrow ID \rightarrow RF \rightarrow X_1 \rightarrow X_2 \rightarrow MEM_1 \rightarrow MEM_2 \rightarrow MEM_3 \rightarrow WB$$

In this new architecture, IF is for instruction fetch; ID for instruction decoding; RF for register file read (of input operands); X_1 and X_2 stages for ALU execution; MEM_1 through MEM_3 for memory access; and WB for write-back. Assume that the branch target address is calculated at stage X_1, and the outcome of the branch condition is determined at stage X_2. Assume further that a store operation finishes at the end of MEM_2 and load finishes at the end of MEM_3.

(a) **{5pts}** Explain what data hazards can happen in this pipeline? What are the possible solutions for data hazards (apart from stalling the pipeline, of course)?
(b) **{5pts}** List two different scenarios illustrating how a control hazard can happen in this architecture.
(c) **{5pts}** If a stalling policy is used, for this 10-stage pipeline, what is the number of stall cycles for a branch instruction? Suggest a way to reduce the number of stall cycles.
(d) **{5pts}** If you want to implement a branch prediction in this architecture, which pipeline stages need to be modified, why, and how?

{AQ5 answer space, continued} {Instruction Pipelining}

OVERFLOW 1 – <u>Will only be graded if explicitly referenced from source problem</u>

OVERFLOW 2 – <u>Will only be graded if explicitly referenced from source problem</u>

SCRATCH PAPER – <u>WILL NOT BE GRADED</u>