Please print in pen:

Waterloo Student ID Number:

WatIAM/Quest Login UserID:

UNIVERSITY OF WATERLOO

Final Exam - Spring 2026 - ECE 350

1. Before you begin, make certain that you have one **2-sided booklet with 16 pages**. You have **150 minutes** to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question.

2. Please read all of the questions before starting the exam, as some of the questions are substantially more time consuming. Read each question carefully. Make your answers as concise as possible. **If there is something in a question that you believe is open to interpretation, then please write your interpretation and assumptions!**

3. All solutions must be placed in this booklet. If you need more space to complete an answer, you may be writing too much. If you need extra space, use the blank space on the last few pages of the exam, clearly label the question there, and note in the original question that you have done so.

Good Luck!

Question	Points Assigned	Points Obtained
1	56	
2	18	
3	12	
4	14	
Total	100	

Q1. (56 points) True/False with Explanation.

For each question:

- Circle True or False and write your explanation below it.
- Explanations must not exceed three sentences.
- One point is awarded for the correct True/False selection.
- One point is awarded for a correct explanation.
- No explanation point is awarded if the True/False selection is incorrect.

1. When a system call passes arguments through a buffer in user memory, the kernel copies them into kernel memory before validating them.

   True False

True: *To avoid time-of-check-to-time-of-use attacks, the kernel must first copy arguments into kernel memory and then validate them. If it validated the data while it still lived in user space, a second user thread could change the values after the check but before use.*

2. Within the same process, one thread cannot access another thread's stack.

   True False

False: *All threads of a process share a single address space. Each thread's stack lives in that shared space, so a thread holding a pointer into another thread's stack can read or write it. Threads are isolated only by convention, not by hardware.*

3. An interrupt handler is a kernel thread that runs at the highest priority.

   True False

False: *Interrupt handlers are not threads and are not scheduled by the scheduler. They run in interrupt context in response to a hardware signal and cannot block like a thread.*

4. I/O requests cannot be completed without first being processed by a device driver.

   True False

False: *A request can be satisfied from a cache. For example, a read() whose data is already in the file-system buffer cache returns from the kernel without involving the device driver or the physical device.*

5. For a fixed number of processes on a uniprocessor, improving throughput will improve the response time of at least one process.

   True False

True: *Suppose response times stayed the same or increased for every process. Then the completion time of the last-finishing process would not improve, so throughput (work completed per unit time) could not improve, a contradiction. Therefore, improving throughput must improve at least one process's response time.*

6. On a multiprocessor, different threads of the same process may run simultaneously on different processors.

   True False

True*: Threads share the address space but execute independently, so the scheduler can run them in parallel on separate processors (which is exactly why cache coherence and synchronization matter).*

7. A Mesa monitor is easier to implement but harder to reason about than a Hoare monitor.

   True False

True: *Mesa semantics (signal-and-continue) are simpler to implement, but a signaled thread is not guaranteed that the condition still holds when it resumes, so it must re-check the condition in a while loop. Hoare semantics (signal-and-wait) hand the lock and a guaranteed condition to the woken thread, which is harder to implement but easier to reason about.*

8. Round-robin (RR) scheduling can be implemented on a system that does not support interrupts.

   True False

True: *The OS can act as an interpreter, executing a fixed number of instructions from each task in turn before switching to the next, achieving round-robin behavior without relying on timer interrupts*

9. If 99% of a program is perfectly parallelizable and there is no parallelization overhead, Amdahl's law predicts a speedup strictly greater than 9.2x on 10 cores.

   True False

False*: By Amdahl's law the speedup is 1 / (0.01 + 0.99/10) = 1 / 0.109 ≈ 9.17, which is less than 9.2. So the claimed 9.2× is not achievable.*

10. In the MSI protocol, if a cache block is in the Modified state in one cache, it must be Invalid in all other caches.

    True False

True: *Modified means that cache holds the only valid, dirty copy with exclusive write permission. The protocol therefore requires every other cache to hold the block in the Invalid state.*

11. Assuming instructions are never written at run time (no self-modifying or dynamically generated code), cache coherence is not needed for instruction caches.

    True False

True: *Under normal execution the CPU only reads instructions and never writes them, so instruction-cache copies do not become stale relative to one another and no coherence protocol is required. (Self-modifying code is the special case the assumption excludes.)*

12. Directory-based coherence protocols are more scalable than snooping protocols.

    True False

True: *Snooping broadcasts coherence traffic on a shared bus, which becomes a bottleneck as the number of cores grows. A directory tracks which caches share each block and sends point-to-point messages only to those caches, so it scales to many more cores.*

13. In a perfectly load-balanced multiprocessor, work stealing is not required.

    True False

True: *Work stealing exists to rebalance load when some cores are idle while others have queued work. If the load is already perfectly balanced, there is nothing to steal, so the (relatively expensive) load-balancing procedure is unnecessary.*

14. With RCU locks, readers that begin reading the shared data during the grace period may or may not see the new data.

    True False

False: *RCU publishes the updated structure before the grace period begins, so a reader that starts during the grace period dereferences the already-updated pointer and sees the new data. The grace period exists only to defer reclaiming the old copy until pre-existing readers finish.*

15. For a workload with total utilization of 1, a rate-monotonic (RM) scheduler cannot produce a feasible schedule.

    True False

False: *Utilization exactly equal to 1 does not by itself doom RM. For certain task sets (for example, periods that are harmonic multiples of one another) RM can still meet all deadlines at U = 1, so the blanket "cannot" is false.*

16. With multi-segment address translation, the same physical address can be mapped to different virtual addresses in two different processes.

    True False

True: *Two processes can assign different segment numbers (hence different virtual addresses) to the same shared physical segment, so a single physical location is reachable through different virtual addresses.*

17. Right after a process is forked, the same global variable in both parent and child has the same physical address if copy-on-write is implemented.

    True False

True: *Copy-on-write initially maps the parent's and child's identical pages to the same physical frames, so before any write the shared global lives at the same physical address. A private copy is made only when one side writes.*

18. Page-table address translation eliminates external fragmentation.

    True False

True: *Pages are fixed size, so any free physical frame can hold any virtual page. There are no variable-size holes to strand, so external fragmentation is eliminated (internal fragmentation is not).*

19. It is possible to protect programs from one another without address translation.

    True False

True: *A simple base-and-limit scheme uses two registers (base and limit) to confine each program to a contiguous physical region, providing protection without full address translation.*

20. For a sparse virtual address space, a multi-level page table is generally more memory-efficient than a single-level page table containing an entry for every virtual page.

    True False

True: *A single-level table needs an entry for every virtual page across the whole space, including large unused gaps. A multi-level table can omit entire lower-level tables for unmapped regions, so it uses far less memory for sparse address spaces.*

21. Consider three periodic tasks, written as (period, execution time), with deadlines equal to periods: {T1: (4, 1), T2: (6, 3), T3: (8, 2)}. An EDF scheduler will generate a feasible schedule.

    True False

True: *For periodic tasks with deadlines equal to periods, EDF is feasible if and only if total utilization ≤ 1. Here the utilization is exactly 1.0, so EDF can schedule the set without missing any deadline.*

22. A two-way set-associative cache can cache two addresses that map to the same cache index.

    True False

True: *A two-way set-associative cache has two ways (lines) per set, so two distinct addresses that share the same index can be held simultaneously, one in each way.*

23. Earliest-deadline-first scheduling can still meet all deadlines for any task set with utilization ≤ 1 when it operates non-preemptively.

    True False

False: *with {(C=1, P=2), (C=3, P=6)}, U = 1, but the second job of the first task misses its deadline because it cannot preempt the 3-unit job.*

24. TLB entries generally include neither an access bit nor a dirty bit.

    True False

False: *If an address translation is cached in the TLB, the page must have already been accessed, so the access bit should already be set. However, the dirty bit is usually maintained in the TLB because, on every write, a page-table walk would otherwise be necessary to set the dirty bit.*

25. The overhead of updating PTEs is the same in a multiprocessor and a uniprocessor.

    True False

False: *A TLB shootdown may be necessary in a multiprocessor system and completes once all relevant processors confirm that the stale entry has been removed. The overhead of a TLB shootdown generally increases with the number of processors involved.*

26. To reduce TLB hit time, TLBs are usually direct-mapped or have low associativity.

    True False

False: *TLBs are typically fully associative. This significantly reduces conflict misses in return for slightly higher hit time.*

27. Demand paging requires a process to be fully resident in physical memory before it can execute.

    True False

False: *Only the pages currently in use need to be resident; the rest can live on disk.*

28. When a keystroke occurs, the kernel schedules an interrupt handler to process it.

    True False

False: *The interrupt handler is invoked directly by the hardware interrupt mechanism; it is not a thread and is not placed on the scheduler's ready queue, so it is not "scheduled."*

Q2. (18 points) Multiple-Select Questions.

1. (2 points) Consider a thread that is executing a busy-wait loop while waiting for another thread. Assume a preemptive scheduler. While the thread is still logically in the busy-wait loop (i.e., it has not yet exited the loop), which of the following may be its scheduler state? Select all that apply.

   ☒ Running
   ☐ Waiting
   ☒ Ready
   ☐ Finished

2. (2 points) Which of the following can be produced by the program below? Select all that apply. Assume no errors occur.

```
void *helper(void *arg) {
    printf("b");
    return NULL;
}

int main() {
    pthread_t thread;
    pthread_create(&thread, NULL, &helper, NULL);
    pthread_yield();
    printf("a");
    return 0;
}
```

   ☒ ab
   ☒ ba
   ☐ b
   ☒ a

3. (2.5 points) Select all correct statements.

   ☒ In a paged address-translation system, the virtual and physical addresses use the same number of page-offset bits.
   ☐ Multi-segment address translation eliminates external fragmentation.
   ☒ Page-table entries in two different processes can map virtual pages to the same physical page frame.
   ☐ An inverted page table always contains fewer entries than the combined multi-level page tables for the same system.
   ☐ The virtual address space is always larger than the physical address space.

4. (2 points) Assume that the address space contains sufficient unused room and that the OS may update the relevant bound, segment-table entries, or page-table entries as the process grows. Which address-translation schemes can represent a process whose heap and stack grow over time? Select all that apply.

☒ Base-and-bound address translation.
☒ Multi-segment address translation.
☒ Single-level page-table address translation.
☒ Multi-level address translation.

5. (2.5 points) Select all correct statements.

☒ Allowing a system-call handler to be preempted at safe points can reduce worst-case scheduling latency in a real-time system.
☒ A real-time task can comprise multiple jobs.
☒ For independent periodic tasks with relative deadlines equal to periods on a fully preemptive uniprocessor, RM is optimal among fixed-priority assignments.
☒ If a polling server is scheduled by RM and all other parameters remain unchanged, reducing its period increases the server's fixed priority.
☒ For independent, fully preemptive jobs on one processor with zero scheduling overhead, EDF and LLF are feasibility-optimal online scheduling policies.

6. (2 points) With the MSI protocol, when is a cache controller forced to write back a cache line B? Select all that apply.

☐ B is in the S state, and another cache requests Get-M for B.
☒ B is in the M state, and another cache requests Get-M for B.
☐ B is in the S state, and it is evicted to make room for another block.
☒ B is in the M state, and it is evicted to make room for another block.

7. (3 points) Which of the following statements about the FAT file system are correct? Select all that apply.

☐ A quick format zeroes all data blocks and marks every FAT entry as free.
☐ FAT supports hard links, so the same file can be reached through several directory entries.
☐ FAT enforces per-user access rights and uses transactional updates to survive crashes.
☐ With 4 KiB blocks, a FAT volume can grow beyond 1 TiB because block addresses are encoded using 32 bits.
☐ FAT provides fast random access to blocks in large files, because each file's block addresses are held in a direct index.
☒ The FAT is stored on disk; the OS may cache FAT sectors in memory, and conventional FAT volumes commonly maintain a second on-disk FAT copy.

8. (2 points) Which of the following statements are **FALSE**? Select all that apply.

☐ Demand paging allows a process to be only partially resident in physical memory.
☐ Fixed-size pages allow the kernel to use any free physical page frame for any virtual page.
☐ Under memory overcommit, paging may avoid the need to swap entire processes out to disk.
☒ Pure segmentation allows the kernel to evict a single segment while keeping the rest of the process running.

Q3. (12 points) Multiple-Choice Calculations.

For each question:

- Select one answer.
- Show your work.
- Two points are awarded for the correct selection.
- Two points are awarded for correct work.
- No work points are awarded if the selection is incorrect.

Consider the following two-level page-table scheme, which is used for the next three questions. The virtual-address, physical-address, and page-table-entry formats are shown below.

Virtual address:

Virtual Page #1	Virtual Page #2	Offset
8 bits	12 bits	12 bits

Physical address:

Physical Page #	Offset
20 bits	12 bits

Page-table entry (PTE):

Physical Page #	OS-defined bits	Unused bits	Control & status bits
20 bits	3 bits	2 bits	7 bits

1. What is the size of a single page? **<u>Show your work</u>**.

☐ 1 KiB
☐ 2 KiB
☒ 4 KiB
☐ 8 KiB

Explanation: *The offset is 12 bits, so a page holds 2^12 = 4096 bytes = 4 KiB.*

2. What is the total maximum size of the page table (in bytes) for this scheme, i.e., the full multi-level structure that maps virtual to physical addresses? Here, "size" means the total number of bytes occupied by entries; ignore page-alignment and allocator padding. **Show your work.**

- ☐ 4194304
- ☐ 4198400
- ☒ 4195328
- ☐ 16777216

Explanation: *Each PTE is 20 + 3 + 2 + 7 = 32 bits = 4 bytes. The top level is indexed by the 8-bit VP#1, giving 2^8 entries = 2^8 × 4 = 2^10 bytes. It points to up to 2^8 second-level tables, each indexed by the 12-bit VP#2, giving 2^12 entries = 2^12 × 4 = 2^14 bytes each, for 2^8 × 2^14 = 2^22 bytes. Total = 2^10 + 2^22 = 1024 + 4194304 = 4195328 bytes.*

3. What is the maximum amount of physical memory addressable with this scheme? **Show your work.**

- ☐ 1 GiB
- ☒ 4 GiB
- ☐ 16 GiB
- ☐ 128 GiB

Explanation: *Each PTE has only 20 bits for physical pages. Therefore, at most there could be 2^20 physical pages, each 2^12 bytes, so the physical memory is 2^20 × 2^12 = 2^32 bytes = 4 GiB.*

Q4. (14 Points) Polling Server, Deferrable Server, and Total Bandwidth Server

Consider the following periodic tasks. Each task has a relative deadline equal to its period and releases its first job at t = 0:

Task	Execution time	Period
T1	1	4
T2	2	8

The following aperiodic jobs each arrive once and have no hard deadline of their own:

Job	Arrival	Execution time
A1	1	1
A2	5	2
A3	12	1

Assumptions:

- Single processor.
- All periodic tasks and aperiodic jobs are fully preemptive.
- Preemption, context-switch, and dispatch costs are zero.
- All events occur at integer time instants.
- Scheduling decisions are made only at integer time instants.
- At any given instant, events are processed in the following order:
- (i) job completions, (ii) server budget replenishments, (iii) periodic-task releases and aperiodic-job arrivals, and (iv) the scheduling decision. Therefore, an aperiodic job arriving exactly at time t is considered pending at t.
- Multiple pending aperiodic jobs are served in FIFO order of arrival: A1 before A2 before A3.
- For the polling server (PS) and deferrable server (DS), the server budget is 2 and the server period is 6 (first budget refill at t = 0), and tasks are scheduled using rate monotonic (RM) policy.
- The total bandwidth server (TBS) uses bandwidth of 1/3.
- EDF tie-breaking rule: the ready entity with the earlier absolute deadline is selected. If two ready entities have the same absolute deadline, the one that became ready earlier is selected. If they are still tied, the periodic task is selected.

1. (2 Points) What are the deadlines assigned to each aperiodic job under TBS. Show your work.

Aperiodic job	Assigned deadline
A1	4
A2	11
A3	15

Explanation: d_k = max(r_k, d_{k−1}) + C_k / U_TBS, with U_TBS = 1/3.
d_1 = max(0,1) + 3 = 4; d_2 = max(4,5) + 6 = 11; d_3 = max(11,12) + 3 = 15.

2. (12 Points) What is the **execution schedule on [0, 16)?** For each unit interval write exactly one of: T1, T2, A1, A2, A3, or idle.

Interval	Polling Server system	Deferrable Server system	TBS system
[0,1)	T1	T1	T1
[1,2)	A1	A1	A1
[2,3)	T2	T2	T2
[3,4)	T2	T2	T2
[4,5)	T1	T1	T1
[5,6)	idle	A2	A2
[6,7)	A2	A2	A2
[7,8)	A2	idle	idle
[8,9)	T1	T1	T1
[9,10)	T2	T2	T2
[10,11)	T2	T2	T2
[11,12)	idle	idle	idle
[12,13)	T1	T1	A3
[13,14)	A3	A3	T1
[14,15)	idle	idle	idle
[15,16)	idle	idle	idle

Explanation: *PS. A1 arrives at 1 and gets scheduled immediately. A2 arrives at 5, but must wait for the polling instant t = 6. At t = 6 neither T1 nor T2 is ready, so the PS runs [6,8): At t = 12 the PS is invoked with queue {A3}, but T1 has the earlier deadline, so the server does not begin until t = 13; it runs [13,14), finishing A3.*
DS. A1 arrives at t = 1; the DS still holds its full budget of 2 and outranks T2, so it preempts T2 and serves A1 in [1,2), then suspends with 1 unit preserved. That preserved unit serves A2 in [5,6); the budget is replenished to 2 at t = 6 (before the dispatch decision), so A2 continues seamlessly in [6,7). At t = 12 T1 outranks the DS, so A3 is served in [13,14).
TBS. The only difference from the DS schedule is at t = 12: A3 has deadline 15 and the fourth job of T1 has deadline 16, so EDF runs A3 first.
Periodic deadline check (all three schedules). T1: jobs complete at 1, 5, 9, and 13 or 14 against deadlines 4, 8, 12, 16. T2: jobs complete at 3 or 4 and at 11 against deadlines 8 and 16. All deadlines are met.