

Please print in pen:

Waterloo Student ID Number:

--	--	--	--	--	--	--	--	--	--

WatIAM/Quest Login UserID:

--	--	--	--	--	--	--	--	--	--

UNIVERSITY OF
WATERLOO



Final Exam - Winter 2023 - ECE 350

1. Before you begin, make certain that you have one **2-sided booklet with 10 pages**. You have **120 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 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. However, if you need extra space, use the blank space on the last page of the exam clearly labeling the question and indicate that you have done so in the original question.

Good Luck!

Question	Points Assigned	Points Obtained
1	22	
2	30	
3	14	
4	12	
5	22	
Total	100	

1. (22 points) True-False with explanation.

For each question:

- Circle your answer and write your explanation below each question.
- Explanations should not exceed 3 sentences.
- One point for correct true-false.
- One point for correct explanation.
- No points for any explanation if true-false is incorrect.

1. In a modern operating system using memory protection through virtual memory, the hardware registers of a memory-mapped I/O device can only be accessed by the kernel.

True False

- False, kernel can map the I/O space of the device into the memory space of a user program

2. Micro-kernel design can improve the resilience of the operating system against bugs.

True False

- True, buggy components are prevented from interfering with other components by the isolation.

3. With only one ready queue for all CPUs on a multiprocessors system, cache reuse is limited.

True False

- True, threads scheduled on different CPUs from the one ready queue, so they can't reliably make use of caches.

4. Threads within the same process can share data with one another by passing pointers to objects on their stacks.

True False

- True, since the threads in the same process share an address space, they can all access the same memory (including each other's stacks).

5. In MSI protocol, if one CPU has a cache block in "Modified" state, that cache block should be in state "Invalid" for the other CPUs.

True False

- True, only one core can be in Modified state at any given time. M (modified) is an exclusive state, forces other CPUs/caches with the data to go to Invalid.

6. The size of an inverted page table is in the order of the number of virtual pages that it translates.

True False

- False, the size of inverted page table is in the order of physical pages that a process uses.

7. Anything that can be done with a monitor can also be done with semaphores.

True False

- True, since monitors can be implemented using semaphores, any monitor-based algorithm can be implemented with semaphores.

8. The lottery scheduling can be used to implement any other scheduling algorithm by adjusting the number of tickets that each process holds.

True False

- False, the lottery scheduling cannot be used to implement strict priority scheduling.

9. A user-level process can modify its own page table entries.

True False

- False, if a user-level process was allowed to modify its own page table entries, then it could access physical memory being used by other processes or the OS kernel. Kernel mode is required to modify page table entries.

10. Interrupt handler is a thread with the highest priority.

True False

- False, interrupt handler is not a thread because it cannot be scheduled.

11. Increasing the size of the memory will always result in reduction in the page-fault ratio regardless of the replacement policy.

True False

- False, with FIFO, it could increase page-fault ratio (Bélády's anomaly).

2. (30 points) Short answers.

1. (3 points) Name three ways in which a processor can transition from user mode to kernel mode.

- (1) Executing a trap instruction for a system call
- (2) Responding to a synchronous exception (e.g., divide by zero)
- (3) Responding to an interrupt

2. (2 points) What needs to be saved and restored on a context switch between two threads in the same process? What if the two threads are from different processes? Be explicit

Within the same process: save/restore all registers, including stack pointer and PC (1 points)
Separate processes: same as "within process" (0.5 point) + save/restore virtual memory info (e.g., page table base register) (0.5 point)

3. (3 points) Which one of the following options can cause a context switch in a system with a non-preemptive scheduler? Select all that apply.

- Calling `yield()` system call.
- Requesting an I/O operation.
- Calling `wait()` operation for another thread (such as `thread-join`).

All three

4. (2 points) What is a monitor?

A lock (Mutex) and zero or more condition variables

5. (2 points) The shortest-remaining-processing-time (SRTF) algorithm requires knowledge of the future. Why is that? How can we approximate the information required to implement SRTF?

SRTF needs to know the future so it can choose the thread with the shortest processing time as the next thread to run. Past behavior can also be used to approximate the future.

6. (2 points) Consider a cache with K cache blocks. Consider a process that repeatedly accesses (K+1) sequential memory blocks. Which cache organization exhibits a higher hit rate with LRU replacement policy? Why?

- Direct mapped cache will have higher hit rate
- Fully associative cache will have higher hit rate
- Both will have equal hit rates

LRU in fully associative cache will result in misses for all accesses.

7. (2 points) Give two reasons why this is a bad implementation for a Mutex:
`mutex.lock() { disable interrupts; }`
`mutex.unlock() { enable interrupts; }`

Possible answers: (1) It prevents hardware events from occurring during the critical section, (2) User programs cannot use this lock, (3) It doesn't work for data shared in multi-processors.

8. (2 points) Which scheduling policy suffers from starvation? Select all that apply.

- First-come-first-served (FCFS) scheduling
- Round-robin (RR) scheduling
- Stride scheduling
- Lottery scheduling (when all threads are assigned at least one ticket)
- Max-min-fair scheduling
- Strict-priority scheduling (when all threads are assigned at least one ticket)
- Multi-level-feedback-queue (MLFQ) scheduling
- Shortest-remaining-time-first (SRTF) scheduling

Strict-priority and SRTF

9. (2 points) Explain how a long-running process can fool the multi-level-feedback-queue scheduler's heuristics into giving it more CPU cycles.

By executing a lot of superfluous I/O, a process could fool or game the scheduler into giving it higher priority by keeping the process in high-priority queues. The net effect would be that the program performing the superfluous I/O might be able to get more time on CPU.

10. (2 points) Explain how the clock algorithm can be modified if hardware does not maintain an access bit for each page-table entry (PTE).

(1) The corresponding PTE for pages that have not been accessed for the first time are set to invalid in the page table. This enables kernel to track the first access through page fault handler.
(2) Once the clock head passes a page that has been accessed in the previous cycle, the corresponding PTE is set to invalid again. This enables kernel to track the next access through page fault handler.

11. (2 points) How does a modern OS regain control of the CPU from a user-level process stuck in an infinite loop?

The timer interrupt handler will enter the scheduler and recover the CPU.

12. (4 points) Name two advantages and two disadvantages of implementing a threading package at user level (i.e., user-managed multi-threading) rather than relying on thread scheduling from within the kernel.

Advantages include fast context switch, low overhead thread creation, user-level synchronization, and user-configurable scheduling. Disadvantages include blocking on I/O and lack of support for multicore.

13. (2 points) Consider the following implementation of a readers-writers lock. Does this implementation prevent starvation of writers? If yes, explain how. If no, explain why.

```
class ReaderWriterLock {
private:
    Mutex mutex;
    CV okToRead;
    Queue<CV> queue;
    int AW = 0;
    int AR = 0;
    int WW = 0;
    int WR = 0;

public:
    void acquireRL();
    void releaseRL();
    void acquireWL();
    void releaseWL();
}

ReaderWriterLock::acquireRL() {
    mutex.lock()
    while (AW + WW > 0) {
        WR++;
        okToRead.wait(&mutex);
        WR--;
    }
    AR++;
    mutex.unlock();
}

ReaderWriterLock::releaseRL() {
    mutex.lock();
    AR--;
    if (AR == 0 && WW > 0)
        queue.head().signal();
    mutex.unlock();
}

ReaderWriterLock::acquireWL() {
    mutex.lock();
    myCV = new CV();
    queue.enqueue(myCV);
    while (AW + AR > 0) {
        WW++;
        myCV.wait(&mutex);
        WW--;
    }
    AW++;
    queue.dequeue();
    mutex.unlock();
}

ReaderWriterLock::releaseWL() {
    mutex.lock();
    AW--;
    if (WW > 0) {
        queue.head().signal();
    } else if (WR > 0) {
        okToRead.broadcast();
    }
    mutex.unlock();
}
```

NO, without tracking positions for writers (numWriters, nextToGo, and myPos), a newly arrived writer thread could steal a signal from a signaled writer.

3. (14 points) **Virtual memory.** Suppose that we have a 64-bit virtual address split as follows:

8 bits [segment ID]	10 bits [table ID]	10 bits [table ID]	10 bits [table ID]	10 bits [table ID]	16 bits [Offset]
------------------------	-----------------------	-----------------------	-----------------------	-----------------------	---------------------

1. (2 points) How big is a page in this system? Explain in one sentence.

Since the offset is 16 bits, the size of a page is $2^{16} = 64\text{KB}$

2. (2 points) How many segments are in this system? Explain in one sentence.

Since there is an 8-bit segment ID, there are $2^8 = 256$ possible segments.

3. (2 points) Assume that the size of each page tables is limited by the size of a single page (so that they can be paged to permanent storage). What is the maximum size of each page table entry (PTE) in this system? Explain in one sentence.

A 2^{16} -byte page with 2^{10} PTEs means that each PTE can be at most $2^{16-10} = 64$ bytes in size

4. (2 points) What bits are required to be included in each PTE for the hardware (i.e., memory management unit (MMU)) to support the 4th-chance algorithm and copy-on-write optimization? **Explain.**

For clock algorithm, we need valid (V), accessed (A), and dirty (D) bits. For copy-on-write, we need Read-only (R) bit.

5. (2 points) A common attack point for Internet worms is to exploit a buffer-overflow bug to execute code on the stack of a victim process. What bit(s) can we add to each PTE to let the MMU prevent this problem and how should we use it (use no more than two sentences)?

Add an execute (E) bit to the PTE. Unless this bit is set, the processor will refuse to execute code in the given page. We prevent the buffer-overflow attack by clearing the Execute bit for any PTEs associated with stack pages.

6. (2 points) Suppose the system has 16 GB of DRAM and that we use an inverted page table instead of a forward page table. What is the maximum number of entries in this table? **Explain.**

$16\text{GB} = 2^{34}$, so the number of physical pages = $2^{34-16} = 2^{18}$, which is the maximum size of each inverted page table

7. (2 points) Without a cache or TLB, how many memory operations are required to read or write a single byte? Explain.

5 actual memory operations: 4 page-table lookups in addition to the actual memory operation. Note that segment table is not maintained in memory.

4. (12 points) Demand Paging. For each of the following page replacement policies, list the total number of page-faults and fill in the contents of the page frames of memory every time its content changes (first three page-faults are marked).

1. (4 points) Min replacement policy:

Reference	A	B	C	D	B	A	B	E	A	D	A	B	A	D	F	
Page 1	A															F
Page 2		B						E				B				
Page 3			C	D												
Mark X for page fault	X	X	X	X				X				X				X

Total number of page faults for Min is 7 (F can be placed anywhere)

2. (4 points) LRU replacement policy:

Reference	A	B	C	D	B	A	B	E	A	D	A	B	A	D	F
Page 1	A			D				E				B			F
Page 2		B								D					
Page 3			C			A									
Mark X for page fault	X	X	X	X		X		X		X		X			X

Total number of page faults for LRU is 9

3. (4 points) FIFO replacement policy:

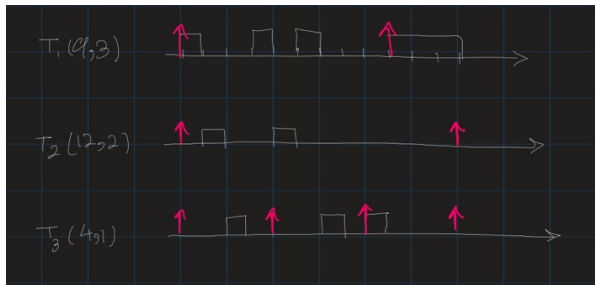
Reference	A	B	C	D	B	A	B	E	A	D	A	B	A	D	F
Page 1	A			D				E				B			
Page 2		B				A				D					F
Page 3			C				B				A				
Mark X for page fault	X	X	X	X		X	X	X		X	X	X			X

Total number of page faults for Min is 11

5. (22 points) Scheduling. Consider three periodic tasks: $T_1(9, 3)$, $T_2(12, 2)$, and $T_3(4, 1)$ running on a uniprocessor system (all unites are in seconds). The first job of all tasks arrives at time 0. Suppose that context switching overhead is zero.

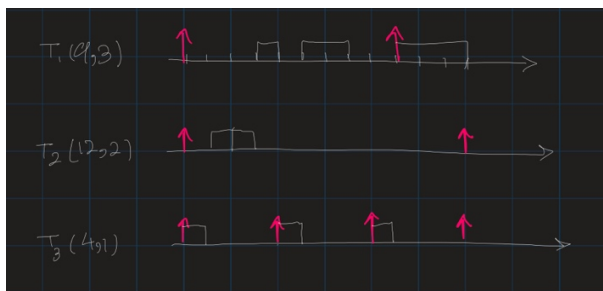
- (6 points) Complete the table below to indicate what the CPU runs for its first 12 seconds under round-robin scheduling with time quantum of 1s. You should think of each job of each task as an independent thread to be scheduled by the scheduler. Use T_{ij} to indicate that CPU runs job j of task i . Use X if CPU idles. Assume that the scheduler always picks jobs of T_1 over jobs of T_2 and T_3 and jobs of T_2 over jobs of T_3 if they arrive to the ready queue at the same time.

Time	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12
Job	T_{11}	T_{21}	T_{31}	T_{11}	T_{21}	T_{11}	T_{32}	X	T_{33}	T_{12}	T_{12}	T_{12}



- (6 points) Complete the table below to indicate what the CPU runs for its first 12 seconds under shortest-remaining-time-first (SRTF) scheduling. You should think of each job of each task as an independent thread to be scheduled. Assume that the scheduler always picks jobs of T_1 over jobs of T_2 and T_3 and jobs of T_2 over jobs of T_3 if they have the same remaining time.

Time	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12
Job	T_{31}	T_{21}	T_{21}	T_{11}	T_{32}	T_{11}	T_{11}	X	T_{33}	T_{12}	T_{12}	T_{12}

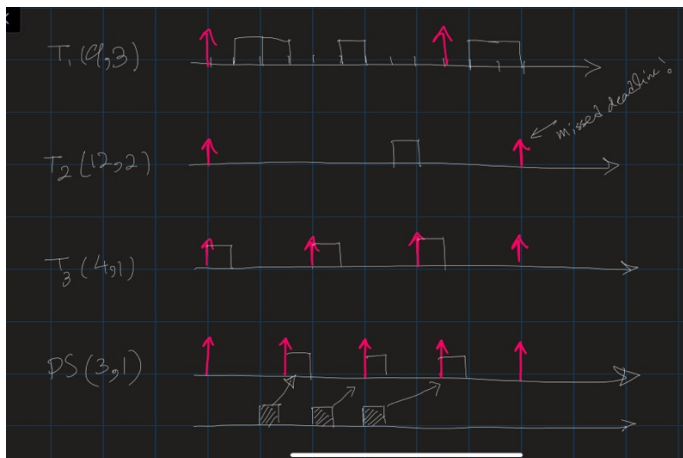


3. (4 points) In addition to T_1 , T_2 , and T_3 , suppose that there are three aperiodic tasks T_4 , T_5 , and T_6 all with execution time of 1 arriving at time 2, 4, and 6, respectively. Complete the table below to indicate what the CPU runs for its first 12 seconds under pooling-server (PS) scheduling where the server's period is 3 and its budget is 1. The server internally runs aperiodic tasks according to a FCFS order. Does any task miss its deadline? **Show your work.**

Time	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12
Job/Task	T_{31}	T_{11}	T_{11}	T_4	T_{32}	T_{11}	T_5	T_{21}	T_{33}	T_6	T_{12}	T_{12}

Missed deadline? If yes, which task?

Yes, T_2



4. (6 points) Consider the same set of periodic and aperiodic tasks as in (3).

a. (1 point) Under total-bandwidth-server (TBS) scheduling, what is the maximum U_{TBS} such that T_1 , T_2 , and T_3 are schedulable? Show your work.

3/12

$$U_{TBS} \leq 1 - 1/3 - 1/6 - 1/4 = 3/12$$

b. (5 points) Complete the table below to indicate what the CPU runs for its first 12 seconds under TBS scheduling with the U_{TBS} that you calculated in (a). If two jobs have the same deadline, scheduler prioritizes the one with lower task number. Show your work.

i. What are a_4 and d_4 ?

2

6

ii. What are a_5 and d_5 ?

6

10

iii. What are a_6 and d_6 ?

10

14

Time	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11	11-12
Job/Task	T_{31}	T_{11}	T_4	T_{11}	T_{32}	T_{11}	T_5	T_{21}	T_{21}	T_{33}	T_6	T_{12}

