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

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

True False

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

True False

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

True False

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

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

True False

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

True False

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

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

True False

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

True False

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

True False

2. (30 points) Short answers.

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

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

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).

4. (2 points) What is a monitor?

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?

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

7. (2 points) Give two reasons why this is a bad implementation for a Mutex:

```
mutex.lock() { disable interrupts; }
mutex.unlock() { enable interrupts; }
```

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

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.

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).

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

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.

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

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

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.

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

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.

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**.

5. (2 points) A common attack point for Internet worms is to exploit a buffer-overrun 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)?

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**.

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

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														
Page 2		B													
Page 3			C												
Mark X for page fault	X	X	X												

Total number of page faults for Min is …….

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														
Page 2		B													
Page 3			C												
Mark X for page fault	X	X	X												

Total number of page faults for LRU is …….

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														
Page 2		B													
Page 3			C												
Mark X for page fault	X	X	X												

Total number of page faults for Min is …….

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.

1. (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												

2. (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												

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												

Missed deadline? If yes, which task?

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.

   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, schedular prioritizes the one with lower task number. Show your work.

      i. What are a_4 and d_4?

      ii. What are a_5 and d_5?

      iii. What are a_6 and d_6?

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												