Please print in pen:

Waterloo Student ID Number:									
WatI	WatIAM/Quest Login UserID:								



Final Exam - Winter 2022 - ECE 350

- 1. Before you begin, make certain that you have one **2-sided booklet with 10 pages**. You have **100 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. 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	40	
2	18	
3	24	
4	18	
Total	100	

1. (40 points) True-False with explanation.

For each question:

- Circle your answer and write your explanation below each question.
- Explanations should not exceed 3 sentences.
- Half a point for correct true-false.
- Half a point for correct explanation.
- No points for any explanation if true-false is incorrect.
- 1. Different threads in the same process share the same heap.

True False

2. Different threads in the same process can access each other's stacks.

True False

3. Efficient implementation of operating system abstractions relies completely on software techniques rather than hardware support.

True False

4. Both type-1 and type-2 hypervisors rely on the host operating system for virtual machine management.

True False

5. CPU utilization is higher in simple-batch operating systems compared to multiprogramming-batch operating systems.

True False

6. For each system call, the operating system can reliably use user-provided arguments if it validates the arguments before copying them from user-space memory to kernel memory.

True False

7. Two user-managed threads within the same process can run simultaneously (in parallel) on two different cores in a multiprocessor.

True False

8. In x86, the user-space stack pointer is saved twice during a mode transfer.

True False

9. In the following code for scheduler functions (used in the lectures for implementation of mutex), interrupts are disabled to guarantee mutual exclusion.

True False

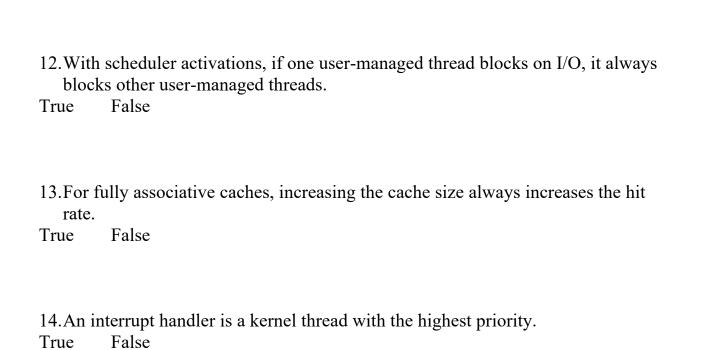
```
Scheduler::suspend(Spinlock *spinlock) {
                                                   Scheduler::make_ready(TCB *tcb) {
       disable_interrupts();
                                                          disable_interrupts();
       scheduler_spinlock.lock();
                                                          scheduler_spinlock.lock();
       spinlock->unlock();
                                                          ready_list.add(tcb);
       runningTCB->state = WAITING;
                                                          thread->state = READY;
       chosenTCB = ready_list.get_nextTCB();
                                                          scheduler_spinlock.unlock();
       thread_switch(runningTCB, chosenTCB);
                                                          enable_interrupts();
       runningTCB->state = RUNNING;
                                                  }
       scheduler_spinlock.unlock();
       enable_interrupts();
}
```

10. Compared to the microkernel architecture, obtaining service in monolithic kernels often requires more mode transfers.

True False

11.On x86 architecture, user programs can execute the instructions cli and sti to enable/disable interrupts.

True False



15. Without atomic load-modify-store instructions, mutual exclusion cannot be implemented in multiprocessors.

True False

16.One of the reasons for BIOS to load bootloader instead of OS is to properly handle multiple OSes.

True False

17. In sequential consistency, the result of any execution is the same as if the operations of all CPUs were executed in a unique total sequential order.

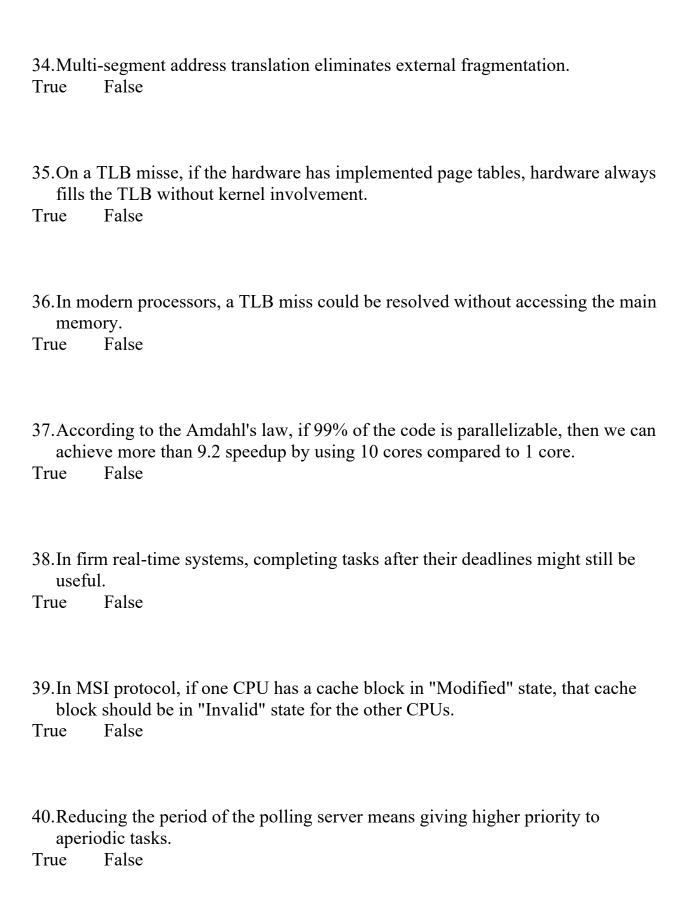
True False

18. Assuming no context-switching overhead, for a fix workload of N tasks, in round-robin scheduling, if task A's CPU burst is shorter than the time quantum, Q, then A's wait time is less than or equal to (N-1)*Q.

True False

19.TLBs True	are typically implemented as fully associative caches. False
	Clock algorithm, if access bit is 1 for all pages every time a page fault ens, then the replacement policy is equivalent to MIN policy. False
21.Each	PCIe device can implement its own address translation cache. False
22.In FA True	T file system, file number is used to index into FAT. False
23.Impro	oving the average response time always improves throughput. False
	non-preemptive scheduler, work-conserving policies always result in lowe ge waiting time compared to non-work conserving policies. False
25.Every True	I/O request will result in the invocation of the device driver's bottom half. False
	"accessed" bit and "dirty" bit can be emulated by the operating system in are instead of being implemented in hardware. False

27.A conflict miss could be a reason for a page fault. True False
28.A shared library code could write data to an absolute virtual address. True False
29.Flash storage pages can be erased individually. True False
30. The size of inverted page table does not provide good cache locality. True False
31. With base and bound address translation, a program with base 0x1100 and bound 0x0100 can access the virtual memory address 0x1110 without raising exceptions True False
32. With multi-segment address translation, the same physical address can be mapped to different virtual addresses in two different processes. True False
33.On a cache miss, caching a whole block of multiple bytes is beneficial because of temporal locality.True False



2. (18 Points) Condition Variables. Fill in the blanks to implement CV using only semaphores. Write at most one statement per line. You may not need all lines.

class CV { private:	signal() {
<pre>public: wait(*mutex) {</pre>	
	}
	<pre>broadcast() {</pre>
}	
	}

- **3.** (24 Points) Fair Scheduling. Consider a uniprocessor system. Suppose that there are 3 tasks, A, B, and C. All three tasks have a CPU burst time of 1 hour, and all three are ready to execute at time 0. Suppose that context switching overhead is zero.
 - a. (12 Points) Suppose that A's weight is 1, B's weight is 2, and C's weight is 3. Complete the table below to indicate what the CPU runs for its first 8 milliseconds under weighted max-min fair scheduling with target latency of 20ms. Use A to indicate that CPU runs task A, B for task B, and C for task C. Assume that the scheduler always picks A over B and C if they have the same virtual time and B over C if they both have the same virtual time. Show your work.

Т	`ime	0 to 1ms	1 to 2ms	2 to 3ms	3 to 4ms	4 to 5ms	5 to 6ms	6 to 7ms	7 to 8ms
T	ask								

b. **(12 Points)** Suppose that A has one ticket, B has 2 tickets, and C has 3 tickets. Complete the table below to indicate what the CPU runs for its first 8 milliseconds under stride scheduling with time quantum of 1ms and W of 600. Use A to indicate that CPU runs task A, B for task B, and C for task C. Assume that the scheduler always picks A over B and C if they have the same pass and B over C if they both have the same pass. Show your work.

Time	0 to 1ms	1 to 2ms	2 to 3ms	3 to 4ms	4 to 5ms	5 to 6ms	6 to 7ms	7 to 8ms
Task								

4. (18 Points) Real-time Systems.

a.	(8 Points) Consider the following tasks: T1(12, 1), T2(6,3), and T3(24,10). Unit
	of time is a millisecond. All tasks arrive at $t = 0$ ms. With EDF scheduling, during
	the first 24 milliseconds, T3 gets preempted by T2 times. (Put a single
	number in the box below - when tasks have the same deadline, A gets higher
	priority, then B, and then C - task A is preempted by task B if a job of B preempts
	an unfinished job of A) Show your work.

- b. **(10 Points)** Consider a uniprocessor with 3 tasks: H, M, and L. Priority of H is high, priority of M is medium, and priority of L is low. Strict-priority scheduling is used to schedule tasks. The system also implements the priority-ceiling protocol.
 - Suppose that there are two mutexes used by different tasks: R2 and R2.
 - L arrives at time t = 0, M arrives at time t = 3ms, and H arrives at t = 6ms.
 - If L runs by itself (without M and H being present in the system), it runs for 1ms. Then, it locks R1. It then runs for 6ms. It then releases R1 and terminates.
 - If M runs by itself (without L and H being present in the system), it runs for 1ms. It then locks R2. Then, it runs for 1ms. Then, it locks R1. Then it runs for another 1ms. Then it releases both mutexes and terminates.
 - If H runs by itself (without L and M being present in the system), it runs for 1ms. It then locks R2. Then it runs for another 2ms. It then releases R2 and terminates.

Complete the table below to indicate what the CPU runs for its first 13 milliseconds. Show your work.

0 to 1	1 to 2	2 to 3	3 to 4	4 to 5	5 to 6	6 to 7	7 to 8	8 to 9	9 to 10	10 to 11	11 to 12	12 to 13