



Examination  
Midterm  
Winter 2024  
ECE 350

Special Materials

Candidates may bring only the listed aids.  
· Calculator - Non-Programmable

Please print in pen:  
Waterloo Student ID Number:

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

WatIAM/Quest Login Userid:

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

Times: Monday 2024-02-26 at 13:30 to 14:30 (1:30 to 2:30PM)  
Duration: 1 hour (60 minutes)  
Exam ID: 5633779  
Sections: ECE 350 LEC 001  
Instructors: Jeff Zarnett

- Instructions:
1. No aids are permitted except non-programmable calculators with no persistent memory.
  2. Turn off all communication devices. Communication devices must be stored with your personal items for the duration of the exam. Taking a communication device to a washroom break during this examination is not allowed and will be considered an academic offence.
  3. Place all bags at the front or side of the examination room, or beneath your table, so they are inaccessible.
  4. There are three (3) questions, each with multiple parts. Not all are equally difficult.
  5. The exam lasts 60 minutes and there are 50 marks.
  6. Verify that your name and student ID number is on the cover page and that your examination code appears on the bottom of each page of the examination booklet.
  7. If you feel like you need to ask a question, know that the most likely answer is “Read the Question”. No questions are permitted. If you find that a question requires clarification, proceed by clearly stating any reasonable assumptions necessary to complete the question. If your assumptions are reasonable, they will be taken into account during grading.
  8. After reading and understanding the instructions, sign your name in the space provided below.

Signature

## 1 Process Management [14 marks total]

### 1.1 Is he dead? Terminated. [5 marks]

ECE 252 covered the subject of signals and introduced the SIGKILL signal which terminates a process immediately without giving it any chance to handle it. This means that the operating system must take action on the SIGKILL. In point form, list the actions that the OS needs to take when a process is terminated suddenly to clean it up.

### 1.2 Process Control Blocks [3 marks]

In the lectures, we've discussed using the Process Control Block to store the registers of a thread when that thread ceases execution. Is it possible to store them somewhere else? Justify your answer.

### 1.3 Concurrency Control [6 marks]

The kinds of locks the operating system implements are advisory, in the sense that they rely on program authors to use them correctly to get the desired outcome. Would it be possible to have mandatory locks, where the OS terminates the program if a shared memory location is accessed without holding the appropriate locks? Explain how this might work, and the tradeoffs.

## 2 Memory [20 marks total]

### 2.1 Virtual Memory [10 marks]

Imagine you have a 32-bit virtual memory system with 4KB pages, as discussed in the lectures, using a translation lookaside buffer, and a regular page table.

1. How would you detect an invalid memory access from a user program (2 marks)?
2. What does the operating system do when a user program does reference an invalid location (1 mark)?
3. What steps take place to transform a logical address into a physical address (3 marks)?
4. What are the tradeoffs in choosing the size of the translation look aside buffer (2 marks)?
5. Would changing to a hierarchical page table with two levels be better? Justify your answer (2 marks).

2.2 Caching [10 marks]

We have a system that has four (4) available frames labelled F0 through F4 that are all initially empty. In the sequence below, each number in square brackets represents an access to that page number.

[8], [7], [3], [1], [5], [3], [9], [5], [7], [6]

Complete both tables below showing the sequence of page replacements that would occur given the sequence above using the replacement strategy printed above the table. To complete each row, show the page in each frame (or write – if it is empty). If a page fault occurs on that access, put Y in the Fault column; put N otherwise. If a fault did occur, increment the total faults by 1; if a fault did not occur then write down the same value for the total faults as the row above.

For the second-chance (clock) algorithm, the *R* bit is reset after every 5 accesses, and will therefore occur once in this table after the 5th access.

First-In-First-Out (FIFO)						Second-Chance (Clock)					
F0	F1	F2	F3	Fault?	Total Faults	F0	F1	F2	F3	Fault?	Total Faults

3 Scheduling [16 marks total]

3.1 Uniprocessor Scheduling [10 marks]

You have a system using round-robin scheduling. At the end of each time slice, the scheduler runs, evaluates the current state, and decides what task to run next. Assume no tasks will ever get blocked for any reason.

The table below gives the breakdown of the tasks. A task that arrives during time slice *n* may be scheduled during time slice *n* + 1 (but not sooner). A task that has an execution length of *k* requires *k* time slices to complete.

Task ID	Arrival Time Slice	Execution Length (time slices)
A	0	4
B	2	3
C	9	5
D	9	2
E	14	3
F	19	1

Each block represents a time slice. In each box below, write the ID of the task that will be executed in that time slice. If there is nothing executing in a time slice, write a dash ( – ) in the box.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

3.2 Sold Out [6 marks]

**Part 1 [3 marks]** Sometimes we have things for which demand vastly exceeds supply, such as concert tickets for a given show. No matter how much the venue might want to sell more tickets, there simply aren't any more seats available. Imagine a similar problem in scheduling: demand for CPU time vastly exceeds the available amount. Even if we use something like FCFS for handling requests, it means some processes may starve. Does this violate the principle of fairness? Justify your answer.

**Part 2 [3 marks]** While the obvious answer of simply adding more capacity might not be possible, an economist would say that the solution is to raise prices (decreasing demand) until the demand and supply are in equilibrium. Think of three (3) strategies that could be applied to reduce the demand for CPU time without adding more capacity.

---

**Extra Space.** Use this for extra space if needed. Be sure to indicate in the original question to refer to the extra space, and indicate here which question is being continued.