Please print in pen:
Waterloo Student ID Number:

WatIAM/Quest Login Userid:

UNIVERSITY OF WATERLOO

Examination
Midterm
Spring 2023
ECE 350

Closed Book

Candidates may bring no aids (no calculators).

Times: Thursday 2023-06-22 at 08:30 to 09:30
Duration: 1 hour (60 minutes)
Exam ID: 5384108
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, some 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

Please initial:

1 Process Management [15 marks]

1.1 Process IDs [6 marks]

Process IDs (PIDs) in UNIX are simple integers and are relatively unique. Consider this code for getting the next:

```
unsigned short get_next_pid( pcb * list_head ) {
  unsigned short i = 2;
  while ( true ) {
      if ( !proc_w_id_exists( i, list_head ) ) {
        return i;
      }
      i++;
  }
}
```

```
bool proc_w_id_exists( unsigned short i, pcb * lh ) {
  pcb * current = lh;
  bool found = false;
  while( current != NULL ) {
    if ( current->pid == i ) {
      found = true;
    }
    current = current->next;
  }
  return found;
}
```

Improvements [4 marks]. Propose two (2) changes that would make this routine more efficient. You do not have to write the actual code, but explain what you would change and why it would help.

Predictability [2 marks]. Another code reviewer says that this makes it possible to predict (with some, but not total certainty) what the next process ID will be. Is that a problem?

1.2 Concurrency Control [4 marks]

When implementing a concurrency control mechanism, the operating system gets to choose which thread is unblocked. If we choose first-in-first-out behaviour, this prevents the possibility of starvation, but ignores priority. A strictly priority-based system is vulnerable to starvation. A colleague proposes a solution: 80% of the time when a thread is to be unblocked, the highest-priority thread is chosen. In the remaining 20% of cases, choose the lowest-priority thread. Analyze this solution: should it be adopted or rejected? Justify your answer.

1.3 Sunrise [3 marks]

Many operating system services run as regular processes, in user mode, rather than in kernel mode or as part of the kernel's internal functionality. Explain why this is the case, as well as the drawbacks for doing so.

1.4 Nap Time [2 marks]

Explain how a system call like `sleep()` can be efficiently implemented using mechanisms we have already discussed (in lectures) around process management.

Please initial:

2 Scheduling: Multicore FCFS [10 marks]

Whether or not we have a multicore CPU, there is always the possibility that there will be multiple current attempts to modify the ready queue which could cause issues. We are familiar already with the idea of compare-and-swap which can be used to manipulate addresses (pointers). For the sake of simplicity, we'll say that we have a First-Come-First-Served scheduling algorithm. Thus, to add an item to the queue you always add it at the end; to remove an item you always take it from the front.

Explain how you would implement the operations of enqueueing and dequeuing the PCBs. Your explanation should explain (1) the data structure you would use; (2) a pseudocode implementation of how you add a PCB to the queue; and (3) a pseudocode implementation of how you remove an item from the front of the queue. The pseudocode must be sufficiently detailed as to explain the algorithm precisely (e.g., you cannot just say "add item to queue"; you would need to explain how to do so in a thread-safe way). Finally, you should also consider cancellation (more explanation below).

1. Data Structure [2 marks]

2. Pseudocode: How to add a PCB [3 marks]

3. Pseudocode: How to remove a PCB [3 marks]

4. Cancellation [2 marks]. Enqueueing and dequeueing are not quite enough, however, because we know that a process can be cancelled at any time. Provide an explanation of how you would deal with this in light of the strategy you have chosen above.

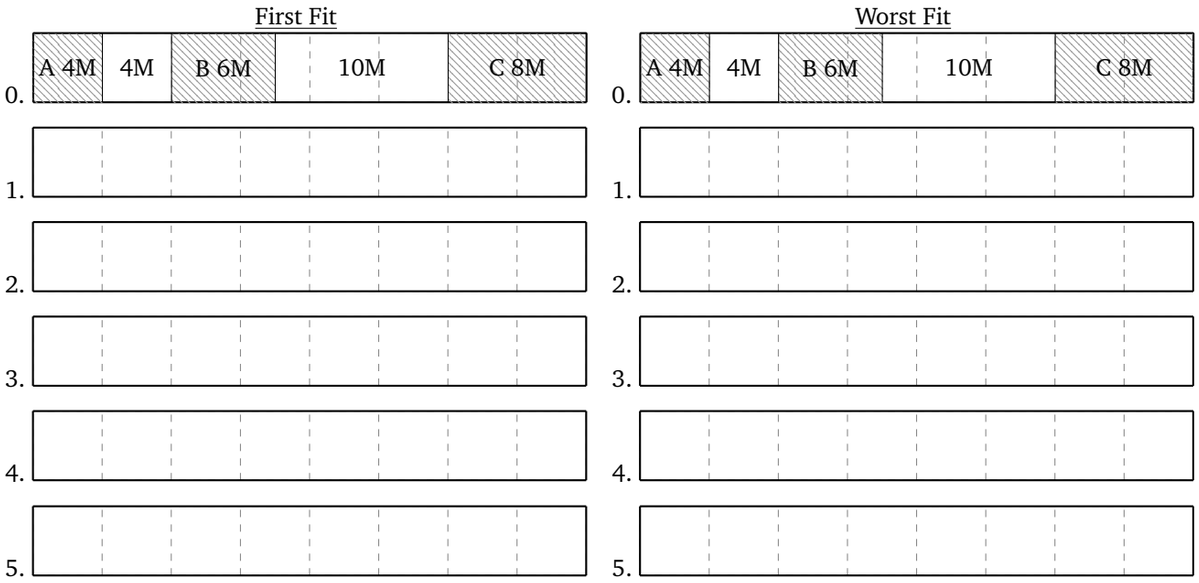
Please initial:

3 Memory [25 marks total]

3.1 Dynamic Memory Allocation [5 marks]

The diagrams below show a 32 MB block of memory used to fulfill memory allocations in this question. Use shading to indicate allocated blocks and also write the size of each block. Grey dashed lines are guidelines in increments of 4 MB to assist you in drawing the blocks in the correct sizes. The initial state of the system is shown as step 0. Perform the allocations and deallocations below using the algorithm. If an allocation cannot be performed, write that in the box for that request and proceed to the next step as in the state of memory was unchanged by the failed request. The most recently allocated block is the 8 MB block. Perform coalescence immediately when possible.

1. Allocate D = 2M
2. Allocate E = 3M
3. Allocate F = 7M
4. Deallocate B
5. Allocate G = 4M



3.2 Bélády's Anomaly [8 marks]

Adding more cache may increase the number of page faults using FIFO cache replacement. This is known as Bélády's Anomaly, named after the researcher who showed its existence. It is possible to demonstrate this when increasing the cache size from 3 to 4 pages, using a sequence of 12 page accesses of pages 1 through 5. Complete the below sequence of page references with a sequence that demonstrates the issue. Show your work in the table.

`[1], [2], [3], [ ], [ ], [ ], [ ], [ ], [ ], [ ], [ ], [ ]`

Page 0	P1	P2	Fault?	Total Faults
1	–	–	Y	1
1	2	–	Y	2
1	2	3	Y	3

Page 0	P1	P2	P3	Fault?	Total Faults
1	–	–	–	Y	1
1	2	–	–	Y	2
1	2	3	–	Y	3

Please initial:

3.3 Instruction and Data Caches [3 marks]

Modern 64-bit processors may have separate level 1 caches for instructions and data. Explain why this is beneficial, making reference to what we know about memory usage patterns.

3.4 Virtual Memory [9 marks]

We have a virtual memory scheme that uses segmentation combined with paging. It has a physical address space of 32 bits and a virtual address space of 32 bits. Page size is 4 KB.

Part 1. Show and explain the layout of a virtual address — that is, the breakdown of the 32 bits into segment and/or page numbers, offsets, etc.

Part 2. For this virtual memory scheme, at some point in time, we have the following page tables for the process currently executing:

Page Table for Segment 0

Page #	Present	Frame #
0	1	3
1	0	–
2	1	8
3	1	4

Page Table for Segment 1

Page #	Present	Frame #
0	1	2
1	1	0
2	1	7
3	0	–

Assuming both page tables are in memory, complete the table below. To do so, determine the physical address corresponding to the virtual addresses. If the address is valid, write "OK" in the third column. If the address is valid but results in a page fault, write "Page Fault" in the third column. If the address is not valid and results in a segmentation fault, put N/A in the physical address and write "Segmentation Fault" in the third column.

Virtual Address	Physical Address	Evaluation
0x010029AA		
0x00001F0C		
0x010049E0		
0x000030D7		

Extra Space. Indicate in the original question to look here, and say here what question you are answering.

Please initial: