PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL 2019 EXAMINATIONS

CSC 369 H1S

Duration: 3 hours

Aids Allowed: None

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read *carefully* every reminder on this page.

- Fill out your name and student number above—do it now, don't wait!
- This final examination consists of 10 questions on 17 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the examination is complete.*
- Answer each question directly on the examination paper, in the space provided, and use a "blank" page for rough work. If you need more space for one of your solutions, use one of the "blank" pages and *indicate clearly the part of your work that should be marked.*
- *Remember that, in order to pass the course, you must achieve a grade of at least 40% on this final examination.*
- As a student, you help create a fair and inclusive writing environment. If you possess an unauthorized aid during an exam, you may be charged with an academic offence.

MARKING GUIDE

N° 1:	_____/10
N° 2:	_____/ 5
N° 3:	_____/11
N° 4:	_____/ 6
N° 5:	_____/ 5
N° 6:	_____/ 9
N° 7:	_____/ 9
N° 8:	_____/ 6
N° 9:	_____/ 6
N° 10:	_____/ 7
TOTAL:	_____/74

STUDENTS MUST HAND IN ALL EXAMINATION MATERIALS AT THE END

Good Luck!

Question 1. Circle the correct answer [10 MARKS]

TRUE FALSE Threads that are part of the same process can access the same TLB entries.

TRUE FALSE The **convoy** effect occurs when longer jobs must wait for shorter jobs.

TRUE FALSE Two processes reading from the same virtual address will access the same contents.

TRUE FALSE A TLB caches translations from virtual page numbers to physical addresses.

TRUE FALSE Threads that are part of the same process share the same page table base register.

TRUE FALSE Deadlock can be avoided by having only a single lock within a multi-threaded application.

TRUE FALSE The number of virtual pages is always identical to the number of physical pages.

TRUE FALSE The OS may not manipulate the contents of an MMU.

TRUE FALSE If a lock guarantees progress, then it is deadlock-free.

TRUE FALSE With dynamic relocation, hardware dynamically translates an address on every memory access.

Question 2. General [5 MARKS]

1. Which of the following components are shared between threads in a multithreaded process? (Check all that apply)

   - ☐ heap memory
   - ☐ stack memory
   - ☐ code
   - ☐ program counter

2. For two processes accessing a shared variable, Peterson's algorithm provides (Check all that apply)

   - ☐ mutual exclusion
   - ☐ bounded waiting
   - ☐ no preemption
   - ☐ progress
   - ☐ preventing circular waiting

3. The program `xxd` outputs the byte sequence, `daa0 2e0e`. This sequence represents an integer in little-endian byte order. Which of the following shows the integer in "normal" order (big-endian)? (Check one)

   - ☐ `2e0e daa0`
   - ☐ `0e2e a0da`
   - ☐ `daa0 2e0e`
   - ☐ `a0da 0e2e`
   - ☐ `e0e2 0aad`

4. `xxd` outputs the following sequence of bytes: `6566 5768`. What string does it represent? (Check one) (Relevant ASCII values: '8' = 56, '9' = 57, 'A' = 65, 'B' = 66, 'D' = 68, 'K' = 75, 'L' = 76, 'V' = 86)

   - ☐ AB9D
   - ☐ D9BA
   - ☐ 9DAB
   - ☐ 8BKV
   - ☐ VKB8

5. Which of the following are stored in an ext2 inode? (Check all that apply)

   - ☐ file size
   - ☐ file name
   - ☐ array of pointers to blocks
   - ☐ type of the file
   - ☐ location of block bitmap
   - ☐ number of links

Question 3. Short Answer [11 MARKS]

Part (a) [1 MARK]

In assignment 1, you made changes to intercept system calls on a virtual machine. Can a student or faculty user do the same thing on teach.cs system. Explain why or why not.

Part (b) [1 MARK]

Explain the purpose of the timer interrupt.

Part (c) [1 MARK]

Instead of a timer interrupt, suppose we have an interrupt that is based on the number of TLB (Translation Look aside Buffer) misses the CPU encounters; once a certain number of TLB misses takes place the CPU is interrupted and OS runs. How would this affect the the usefulness of the interrupt?

Part (d) [2 MARKS]

Explain the steps needed to read the first byte of the file `/A/B/c.txt` in an ext2 file system. Assume nothing is cached in memory and state any other assumptions you need to make.

Part (e) [1 MARK]

What data structure allows the kernel to determine when a process is accessing an invalid memory area?

Part (f) [1 MARK]

In round-robin scheduling, new processes are placed at the end of the queue, rather than at the beginning. Suggest a reason for this.

Part (g) [2 MARKS]

The multi-level feedback queue (MLFQ) policy periodically moves all jobs back to the top-most queue. On a particular system this is done every 10 seconds. Consider the effects of shortening this time interval to 1 second.

i– [1 MARK] Explain a positive effect that may occur.

ii– [1 MARK] Explain a negative effects that may occur (other than increased overhead costs).

Part (h) [2 MARKS]

Is the following statement true or false? Explain your answer. *A longer scheduling time slice is likely to decrease the overall TLB miss rate in the system.*

Question 4. Memory [6 MARKS]

Part (a) [2 MARKS]

Malloc is often implemented using the same approach as dynamic partitioning. Suppose we have a very simple implementation of malloc in which we always allocate new space at the end of previously allocated memory. This means that holes that result from calls to `free` will not be reused until the entire address space fills up. This is feasible since most programs never use their full virtual address space.

Identify and explain one undesirable property of this algorithm.

Part (b) [1 MARK]

Now suppose that we improve the algorithm by looking for appropriately sized holes from previous calls to `free` when we need to allocate new space. Only when an appropriately sized hole does not exist will we allocate new space at the end of the previously allocated space. Identify and explain one problem with this approach.

Part (c) [1 MARK]

An alternate approach to implementing `malloc` would be to maintain a list of free blocks of varying sizes. Then a request for space from `malloc` would be given the smallest size block that was big enough to satisfy the request. When a block is freed, it is returned to the appropriate list of free blocks.

Describe one advantage that this algorithm has over the one described in Part (a)?

Part (d) [2 MARKS]

What factors would influence design decisions on the sizes of the data blocks that `malloc` should use in implementing the algorithm described in Part (c)?

Question 5. Synchronization [5 MARKS]

Consider the following code for inserting into a shared linked list. Assume that `malloc` always succeeds. Only the line numbers of interest are labelled. If thread T executes line 1 we label it as T1. If the scheduler is run such that thread T executes the first line of the function and then thread S executes the first line of the function, we would write the schedule as T1 S1

```
struct node {
    int key;
    struct node *next;
}

void insert(struct node **head, int key){
1   struct node *new = malloc(sizeof(struct node));
2   new->key = key;
3   new->next = *head;
4   *head = new;
}
```

In each of the subquestions below, assume we have a variable L of type `struct node *` that points to a list two nodes with keys 3 and 4. Assume thread T calls `insert(&L, 2)` and thread S calls `insert(&L, 5)`. (Each question has the same initial state.)

Part (a) [1 MARK]

If the two threads run according to the following schedule, check the box that indicates the correct final state of the list: T1 T2 T3 T4 S1 S2 S3 S4

- ☐ 2 5 3 4
- ☐ 5 2 3 4
- ☐ 2 3 4 5
- ☐ 2 3 4
- ☐ 5 3 4

Part (b) [2 MARKS]

Write a schedule using the notation described above where the final state of the list is 2 3 4. Assume both threads run correctly to completion.

Part (c) [2 MARKS]

Add appropriate synchronization to the insert method to prevent the race condition illustrated above. Full marks only for solutions that maximize concurrency.

Question 6. Synchronization [9 MARKS]

At a sports centre, there is one party room where fans can gather to cheer on their team. However, fans of different teams don't get along, so only fans of the same team can be in the room at the same time. If there is one fan of team BLUE in the room then only fans of team BLUE may enter. Fans of team GOLD may not enter until all the fans of team BLUE have left. When the room is empty a fan of either team may be the first one to enter the room.

This problem is simulated with one thread for each fan. A fan is represented the `struct fan`.

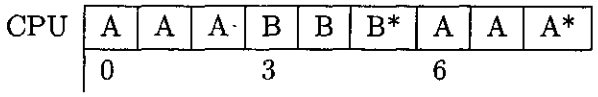
```
enum team_id { BLUE, GOLD, NONE };

struct fan {
  int id;
  enum team_id team;
};

void enter_room(struct person *p);
void leave_room(struct person *p);

/* This is the main function that a thread runs when it is created. There is
 * one thread for each fan, and the struct fan has been correctly initialized
 * with a team_id.  You may not change this function */

void *go_to_party(void *me) {
    struct fan *p = me;
    enter_room(p);
    // Enjoy the party for a period of time
    leave_room(p);
    return NULL;
}
```

Your task is to write the functions `enter_room` and `leave_room` using appropriate synchronization to achieve the specifications above. You may declare any global variables necessary and do not need to initialize synchronization variables.

Question 6. (CONTINUED)

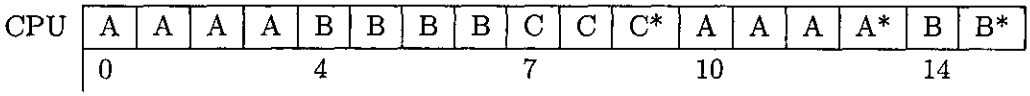
Question 7. Scheduling [9 MARKS]

For this question we will use diagrams to depict scheduling algorithms. For example, let's say we run job A for 3 time units, and then run job B for 3 times units and then run job A again for 3 time units. A diagram of this schedule is shown below. Note that an * is placed on the last execution unit of a process indicating that it is done and will not run again.



Part (a) [2 MARKS]

Assume the following schedule for a set of three jobs: A, B, and C.



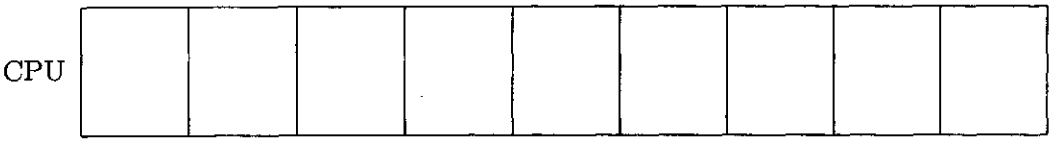
Which of the scheduling disciplines below could allow this schedule to occur? Check a box if it **is** possible for the scheduling policy to lead to the schedule above.

- ☐ Round Robin (RR)
- ☐ First In First Out (FIFO)
- ☐ Shortest Remaining Time First (SRTF)
- ☐ Multi-level Feedback Queue (MLFQ)

Part (b) [2 MARKS]

Complete the picture of the non-preemptive shortest job first (SJF) scheduling with three jobs that arrive at the same time:

- A - 4 time units
- B - 2 time units
- C - 3 time units

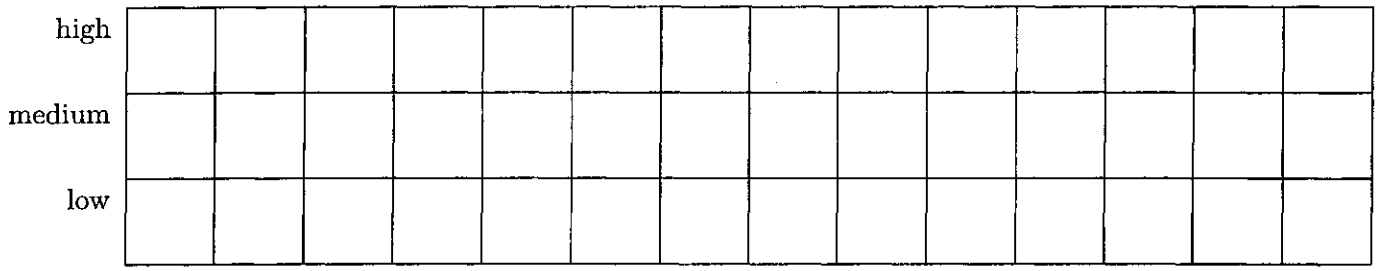


Part (c) [5 MARKS]

Assume we have a multi-level feedback queue (MLFQ) scheduler with three levels of priority. The time-slice (quantum) for jobs with highest priority is 1 time unit, for jobs with the middle priority is 2 time units, and 3 for the lowest priority.

i– [2 MARKS] What is the benefit of assigning a smaller-time slice for higher priority jobs? Explain your answer.

ii– [3 MARKS] Assume jobs A and B arrive at the same time (A arrives just before B). They each have a run-time of 7 time units and are CPU-bound, making no I/O requests. Using the properties of the MLFQ scheduling policy as above, draw a picture of how the scheduler behaves. The queues are labeled below.

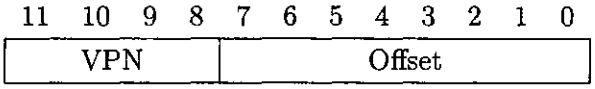


Question 8. Address Translation [6 MARKS]

Consider the linear page table configuration below.

Remember: Each hexadecimal digit is 4 bits in binary.
$(2^4 = 16, 2^6 = 64, 2^8 = 256, 2^{10} = 1024, 2^{12} = 4096, 2^{14} = 16,384)$

- Virtual address format: 4-bit virtual page number, 8-bit offset



- Page Table Entry (PTE) size: 2 bytes
- PTE format: High order bit is the valid bit. The lowest 6 bits are the Page Frame Number

Part (a) [3 MARKS]

Fill in the values in the table below assuming all the bits are used in the virtual address and page frame numbers. Use decimal (base 10) in your answers. Sizes should be expressed in bytes.

Page size (virtual and physical)	
Virtual address space size	
Physical memory size	

Part (b) [3 MARKS]

Consider the following the page table.

VPN	Entry
0	0x8005
1	0x0000
2	0x8013
3	0x8015
4	0x0000
5	0x8012
6	0x8009
7	0x801f
8	0x0000
9	0x800d
10	0x801d
11	0x0000
12	0x800e
13	0x8010
14	0x800c
15	0x0000

For each virtual address, write down the physical address that it maps to, or invalid if it does not map to a valid physical address.

VADDR 0x0ca6	
VADDR 0x09a3	
VADDR 0x0195	

Question 9. Page Replacement [6 MARKS]

In this question, we look at a string of memory references to virtual pages, and we are told what happened on each reference: whether a particular memory reference was a Translation Lookaside Buffer (TLB) hit, TLB miss, or a page fault.

Part (a) [2 MARKS]

Here is the first trace, from some system:

Input	Output
load to virtual page 0	Page fault
load to virtual page 1	Page fault
load to virtual page 2	Page fault
load to virtual page 3	Page fault
load to virtual page 0	TLB miss
load to virtual page 1	TLB miss
load to virtual page 2	TLB miss
load to virtual page 3	TLB miss

Assuming a TLB with LRU-based replacement (Least Recently Used), what do we know about size of the TLB? (where size is number of entries the TLB holds)

Part (b) [2 MARKS]

On a different system, we see this trace:

Input	Output
load to virtual page 0	Page fault
load to virtual page 1	Page fault
load to virtual page 2	Page fault
load to virtual page 3	Page fault
load to virtual page 0	TLB hit
load to virtual page 1	TLB hit
load to virtual page 2	TLB hit
load to virtual page 3	TLB hit

Again assuming a TLB with LRU-based replacement, what can we conclude about the size of the TLB?

Part (c) [2 MARKS]

Finally, on some third system we see the following stream:

Input	Output
load to virtual page 0	Page fault
load to virtual page 1	Page fault
load to virtual page 2	Page fault
load to virtual page 3	Page fault
load to virtual page 0	TLB hit
load to virtual page 1	TLB hit
load to virtual page 2	Page fault
load to virtual page 3	TLB miss

Assume the TLB can hold two entries, and the entire memory can hold three page frames. What can we conclude about the replacement policy used by the OS?

Question 10. File system consistency [7 MARKS]

On an ext2 file system, consider the operation of creating a new empty file in an existing directory. Assume the directory occupies one block and there is enough space to add a new entry. Assume the directory inode and the file inode are in different disk blocks.

Which of the following blocks must be updated? Check all that apply.

- ☐ inode bitmap
- ☐ data bitmap
- ☐ file inode
- ☐ directory inode
- ☐ directory data block

In each of the remaining questions, check the box that most closely explains what happens if a crash occurs after updating only the block(s) specified.

Bitmap

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

File inode

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

Directory inode and directory data

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

Bitmap and file inode

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

Bitmap and directory inode and directory data

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

File inode and directory inode and directory data

- ☐ No inconsistency (it simply appears that the operation was not performed)
- ☐ Data leak (data block is lost for any future use)
- ☐ Inode leak (Inode is lost for any future use)
- ☐ Multiple file paths may point to same inode
- ☐ Something points to garbage
- ☐ Multiple of the problems listed above

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.