University of Waterloo
CS350 Midterm Examination

Winter 2018

Student Name: _____

Closed Book Exam
No Additional Materials Allowed

This page intentionally left blank.

1. (12 total marks) True or false.

a. _____ MIPS treats all interrupts and exceptions as exceptions.

b. _____ When implementing a spinlock, **lw** and **sw** are used to test-and-set the lock atomically.

c. _____ It is not possible to have consecutive trapframes on a stack.

d. _____ A semaphore can allow multiple threads into a critical section.

e. _____ The bits used for the segment number limit the size of the segment.

f. _____ Volatile variables guarantee atomicity of loads and stores.

g. _____ System calls are exceptions in MIPS.

h. _____ All processes terminate by a call to **_exit**.

i. _____ Dynamic relocation is efficient with respect to translation space and time.

j. _____ Address spaces contain but do not use $0x0$ because this address is reserved for NULL.

k. _____ Paging eliminates internal fragmentation.

l. _____ Paging eliminates external fragmentation.

This page intentionally left blank.

2. (14 total marks) Short Answer

a. (3 marks) List three possible sources of race conditions.

b. (2 marks) On a multi-CPU system, suppose interrupts are turned OFF and no re-arranging of code is performed. Is this sufficient to prevent race conditions without using synchronization primitives? Why or why not?

c. (2 marks) Suppose a program has a global array of N items. N threads are forked such that thread i only reads/writes array element i. The global array is not used by any other threads. Is synchronization required for this global array? Why or why not?

This page intentionally left blank.

d. (5 marks) List the five steps required to implement **cv_wait**.

e. (2 marks) How are binary semaphores different from locks?

This page intentionally left blank.

3. (4 total marks)

A process was running user code on the CPU when an interrupt was received. The interrupt is for **waitpid**. While executing **sys_waitpid** there is a timer interrupt. Draw the user and kernel stacks for this process up to and including the point of executing **timer_interrupt_handler**.

This page intentionally left blank.

4. (5 total marks)

Consider the following pseudo code:

```
Queue carBuffer[N];
Semaphore parkingSpots( N );
Semaphore cars( 0 );

MakeCar()
{
    P( parkingSpots );
        carBuffer.add();
    V( cars );
}

TakeCar()
{
    P( cars );
        carBuffer.pop();
    V( parkingSpots );
}
```

a. (1 marks) Is there a race condition in this code?

b. (4 marks) If you answered (a) yes, fix the code. Otherwise, explain why there is no race condition.

This page intentionally left blank.

5. (9 total marks)

a. (6 marks) Consider an implementation of memory segmentation. What would be required to support growing segments? List the steps, remember that there is a maximum segment size.

b. (3 marks) What are the problems associated with segmentation?

This page intentionally left blank.

6. (12 total marks)

A system uses 64 bit physical addresses and 16 bit virtual addresses. Frame and page size is 4KB (2^{12}bytes).

a. (1 mark) How many bits are required for the page offset?

b. (1 mark) How many frames of physical memory are there?

c. (1 mark) How many pages of virtual memory are there?

d. (1 mark) How many bits are required for the frame number?

e. (1 mark) How many bits are required for the virtual page number?

f. (1 mark) Suppose each process used the maximum amount of virtual memory, and, on-demand paging is not used. What is the maximum number of process that could live in physical memory at the same time? (Assume the kernel occupies 0 bytes).

g. (1 mark) If a process uses 1KB (2^{10}bytes) of memory for its address space, how much memory is wasted due to internal fragmentation?

This page intentionally left blank.

Question 6 continued.

h. (4 marks) Which of the following virtual addresses are valid for a process that uses the first $0 \rightarrow 2^{10} - 1$ bytes of virtual memory?

i. 0x0000

ii. 0xFEE5

iii. 0x1234

iv. 0x010A

i. (1 mark) If each entry in the page table is 2^4 bytes, what is the size of the page table?

This page intentionally left blank.

7. (6 total marks)

A program has N global queues of equal length M. Suppose we wish to create a function called **AddElement(queue a, queue b, int i)** which adds the i^{th} element of queue **a** to thei^{th} element of queue **b**. Note that **a** and **b** must be unique queues.

We want to implement synchronization for **AddElement** such that as many threads as possible may execute **AddElement** concurrently.

A throwaway on reddit suggested that you use N locks—one for each queue and acquire them in increasing order of queue number.

```
void AddElement( queue a, queue b, int element )
{
        if ( a.num < b.num )
        {
                acquire( a.lock );
                acquire( b.lock );
        }
        else
        {
                acquire( b.lock );
                acquire( a.lock );
         }
  }
```

a. (1 mark) What is the maximum number of threads that can execute **AddElement** concurrently (without a race condition)?

b. (1 mark) Is there a solution that lets more threads execute **AddElement** concurrently (without a race condition)?

c. (4 marks) If yes, describe that solution. If no, explain why.

This page intentionally left blank.

8. (6 total marks)

Your employer asks you to implement **bool try_acquire(lock * lk)** for locks. This function returns **true** and takes the lock if it is available, or returns **false** if the lock is not available. **try_acquire** does not force the calling thread to block if the lock is not available.

List the steps to implement **try_acquire**.

This page intentionally left blank.

9. (8 total marks)

A system uses 32bit physical and virtual addresses and memory segmentation. There are 4 segments, two bits are used for the segment number. The relocation and limit for each of the 4 segments are:

Segment #	Relocation	Limit
0	0x1000 0000	0x1000
1	0x8000 0000	0x4000
2	0x3400 0000	0x0200
3	0xA000 1000	0xA000

Translate the following addresses from virtual to physical. Clearly indicate what segment each address belongs to.

0x0EA5 EE00

0x0000 0ACE

0x3000 00C5

0x2000 AFAF