University of Waterloo
CS350 Midterm Examination

Spring 2017

Student Name: _____

Closed Book Exam
No Additional Materials Allowed

This page intentionally left blank.

1. (12 total marks) True or false. If false, justify your answer.

a. _____ Each thread has its own address space but can share global variables with other threads.

b. _____ A binary semaphore, i.e., a semaphore with one resource, is identical to a lock in every way.

c. _____ MIPS instructions **ll** and **sc** provide sufficient atomicity to implement a spinlock.

d. _____ It is generally better to block than to spin.

e. _____ User processes have two address spaces; one for the user data and one for the kernel data.

f. _____ While the OS/161 kernel is handling a system call, such as **fork**, it is possible to be preempted.

g. _____ A TLB miss raises an exception which is handled by **vm_fault**.

This page intentionally left blank.

h. _____ A single level page table must fit into a single page of virtual memory.

i. _____ For single and multi-level paging it is most efficient for page size to be equal to frame size.

j. _____ The kernel does not distinguish between ready threads and ready processes during scheduling.

k. _____ The number of processes that can run concurrently is primarily determined by the amount of physical memory and the virtual memory implementation, not the performance of the CPU.

l. _____ The MMU for dynamic relocation must have a relocation and limit register for every process.

This page intentionally left blank.

2. (11 total marks) Short Answer

a. (2 marks) Process A calls fork once producing a single child process, B, with PID 4765. After foking process 'B', 'A' then calls **waitpid** on PID 4765, then immediately calls **waitpid** on 9382. Does the second call to waitpid succeed? Why or why not?

b. (2 marks) In the implementation of **lock_acquire**, the wait channel must be locked prior to releasing the spinlock. Why?

c. (2 marks) A global shared variable is read, but not written to, by multiple threads. Is a lock required? Why or why not?

This page intentionally left blank.

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

This page intentionally left blank.

3. (8 total marks)

A user process calls **getpid**. Before returning from **sys_getpid** there is a timer interrupt that causes a context switch. Draw the user and kernel stacks for the process at this exact time.

This page intentionally left blank.

4. (8 total marks)

Consider function **increment_max_arrays(queue * a, queue * b, queue * c)**:

```
void increment_max_arrays( queue * a, queue * b, queue * c )
{
    for ( int i = 0; i < a->length(); i ++ )
    {
        if ( a[i] > b[i] && a[i] > c[i] )
            a[i] ++;
        else if ( b[i] > a[i] && b[i] > c[i] )
            b[i] ++;
        else
            c[i] ++;
    }
}
```

For example, if $a = [0, 1, 2]$, $b = [2, 3, 1]$, and $c = [8, 0, 0]$, then after calling **increment_max_arrays**: $a = [0, 1, 3]$, $b = [2, 4, 1]$, and $c = [9, 0, 0]$. The function **increment_max_arrays** can be run concurrently. Suppose there are N queues of equal length and N locks — one for each queue.

a. (4 marks) Give a pseudocode implementation of the synchronization required for **increment_max_arrays** using the "No Hold and Wait" strategy for deadlock prevention. You may assume the lock function **bool try_acquire(struct lock * lck)** exists. If the lock is available **try_acquire** acquires the lock and returns true. If the lock is not available, the function returns false.

This page intentionally left blank.

b. (4 marks) Give a pseudocode implementation of the synchronization required for **increment_max_arrays** using "Resource Ordering".

This page intentionally left blank.

5. (11 total marks)

A system uses 32 bit physical addresses and 24 bit virtual addresses. Frames and pages are 1KB (2^{10} bytes).

a. (1 mark) What is the maximum amount of addressable physical memory?

b. (1 mark) What is the maximum amount of addressable virtual memory?

c. (1 mark) How many page table entries (PTEs) are in the page table?

d. (1 mark) How many bits are needed for the page number?

e. (1 mark) How many bits are needed for the page offset?

f. (1 mark) If each PTE is 8 bytes (2^3), how big is the page table?

This page intentionally left blank.

A system uses 32 bit physical addresses and 24 bit virtual addresses. Frames and pages are 1KB (2^{10}).

g. (5 marks) A process has a 16KB (2^{14} bytes) address space. By coincidence, frame number is equal to page number plus 0x3.

i. (1 mark) How many pages are valid in the page table?

ii. (1 mark) What is the physical address for virtual address 0x$A5A5A5$?

iii. (1 mark) What is the physical address for virtual address 0x16?

iv. (1 mark) What is the virtual address for physical address 0x14$A6$?

v. (1 mark) What is the virtual address for physical address 0x10000000?

This page intentionally left blank.

6. (8 total marks)

Consider the following code:

```
int balance = 0;

void Deposit( int value ) {
    balance += value;
}

void Withdrawl( int value ) {
    balance -= value;
}

void Thread1( void * n, unsigned long tNum ) {
    for ( int i = 0; i < tNum; i ++ ) {
        Deposit( tNum );
    }
}

void Thread2( void * n, unsigned long tNum ) {
    for ( int i = 0; i < tNum; i ++ ) {
        Withdrawl( i );
    }
    thread_fork( "deposit", NULL, Thread1, NULL, 4 );
}

int main() {
    thread_fork( "deposit", NULL, Thread1, NULL, 4 );
    for ( int i = 0; i < 4; i ++ ) {
        thread_fork( "withdrawl", NULL, Thread2, NULL, 4 );
    }
}
```

a. (1 mark) What is the total number of threads?

b. (1 mark) This code contains a race condition. If **balance** was **volatile** would it provide sufficient synchronization to prevent the race condition?

This page intentionally left blank.

c. (3 marks) Suppose the main thread needs to wait for all other threads to exit before continuing. Describe the changes required to implement this behaviour.

d. (3 marks) Which of the following are possible final values for **balance**?

16

18

-16

-18

0

34

This page intentionally left blank.

7. (10 total marks)

A 32 bit virtual memory implementation combines both paging and segmentation. Each address space is comprised of three segments: code, data and stack. Each segment has its own page table and has a maximum size of 2^{20} bytes. The MMU will determine both the segment and page number to perform the translation.

a. (1 mark) The highest order bits of the virtual address are reserved for the segment number. How many bits are needed?

b. (3 marks) What registers does the MMU need?

c. (2 marks) How does the MMU determine if a virtual address is valid?

d. (2 marks) What must the kernel do during a process context switch?

e. (2 marks) If the code segment must be read-only, what changes are needed to support this feature?

This page intentionally left blank.

8. (12 total marks)

The system call **kill** is used to terminate a process. **kill** takes one parameter, the PID of the process to terminate. If no process exists with that number, **kill** returns an error.

a. (5 marks) List the steps required to implement **sys_kill**, including any changes to data structures that might be needed.

This page intentionally left blank.

In many operating systems you can only kill a process that you created, or, if you have root/admin access.

b. (2 marks) What additional data must the kernel store to implement this feature?

c. (1 mark) What additional data must the process store to implement this feature?

d. (4 marks) List any additional changes to **sys_kill** that may be required to implement this feature.

This page intentionally left blank.