Name	**UBID**	**Seat**	**Side**

Question:	1	2	3	4	5	6	7	8	Total
Points:	10	5	5	5	5	5	5	20	50
Score:									

CSE 421/521 Midterm Exam

26 Mar 2014

This midterm exam consists of three types of questions:

1. **10 multiple choice** questions worth 1 point each. These are drawn directly from lecture slides and intended to be easy.
2. **6 short answer** questions worth 5 points each. You can answer as many as you want, but we will give you credit for your best four answers for a total of up to 20 points. You should be able to answer the short answer questions in four or five sentences.
3. **2 long answer** questions worth 20 points each. **Please answer only one long answer question.** If you answer both, we will only grade one. Your answer to the long answer should span a page or two.

Please answer each question as **clearly** and **succinctly** as possible. Feel free to draw pictures or diagrams if they help you to do so. **No aids of any kind are permitted.**

The point value assigned to each question is intended to suggest how to allocate your time. So you should work on a 5 point question for roughly 5 minutes.

Please fill out your name and UB ID number above. Also write your UB ID number at the bottom of each page of the exam in case the pages become separated.

There are **three** scratch pages at the end of the exam if you need them. If you use them, please clearly indicate which question you are answering.

I have neither given nor received help on this exam.

Sign and Date: ______________________________

Multiple Choice

1. (10 points) Answer all **ten** of the following questions. Each is worth **one** point.

   (a) What song was played *twice* before class this semester?
   ◯ "Breathing Underwater" by Metric. ◯ "Ready to Start" by Arcade Fire. ◯ "The Hockey Song" by Stompin' Tom Connors. ◯ This is a trick question; GWA never plays a song twice!

   (b) Which of the following instructions can cause an exception?
   ◯ All of them. ◯ `addiu` ◯ `lw` ◯ `syscall`

   (c) Con Kolavis was particularly interested in improving what aspect of Linux scheduling?
   ◯ Overhead. ◯ Throughput. ◯ Interactive performance. ◯ Awesomeness.

   (d) What interface does something have to support to look like memory?
   ◯ `lock()` and `unlock()`. ◯ `malloc()` and `free()`. ◯ `fork()` and `exec()`. ◯ Load and store.

   (e) Acquiring a lock will never create a synchronization problem.
   ◯ True. ◯ False.

   (f) Which of the following is *not* a requirement for deadlock?
   ◯ Multiple independent resource requests. ◯ A linear dependency graph. ◯ Protected access to shared resources. ◯ No resource preemption.

   (g) Threads can be implemented without kernel support.
   ◯ True. ◯ False

   (h) Which of the following is the simplest scheduling algorithm to implement?
   ◯ Rotating staircase. ◯ Multi-level feedback queue. ◯ Round-robin. ◯ Random.

   (i) Which of the following is *not* an example of an operating system mechanism?
   ◯ A context switch. ◯ Loading a virtual address into the TLB. ◯ Locks and semaphores. ◯ Prioritizing interactive threads.

   (j) All of the following are a good fit for the address space abstraction *except*
   ◯ virtual-to-physical address translation. ◯ `sbrk`. ◯ the TLB. ◯ base-and-bounds address translation.

UB ID:

Short Answer

Choose **4 of the following 6** questions to answer. You may choose to answer additional questions, in which case you will receive credit for your best four answers.

2. (5 points) We have discussed several cases where operating systems provide a useful illusion to processes. Name one, describe why it is useful, and briefly explain how it is provided, identifying any hardware cooperation required.

3. (5 points) First, explain how locking shared data structures can reduce performance on multicore systems. Second, describe how to perform more intelligent locking to improve the performance of the following code snippet *without* sacrificing correctness or significantly increasing the amount of space needed to store the items. As a hint, you can assume that the loop frequently has to examine many entries before it finds one that is available. The code to remove entries is not shown, but you can assume that entries are periodically removed.

```c
struct item {
  bool valid;
  int value;
};

struct item array[32768];
struct lock * arrayLock;

int
saver(int doubleRainbow)
{
  bool failed = 1;

  // Assume that arrayLock was properly initialized.

  lock_acquire(arrayLock);

  for (int i = 0; i < 32768; i++) {
    if (array[i].valid == 0) {
      array[i].value = doubleRainbow;
      array[i].valid = 1;
      failed = 0;
      break;
    }
  }

  lock_release(arrayLock);
  return failed;
}
```

UB ID:

Extra space for Problem #3

4. (5 points) Identify three system calls that allocate new virtual addresses (i.e., allow the process to use them) and describe the semantics associated with each.

5. (5 points) So far we have discussed memory as being uniform, meaning that from the perspective of the core (or cores) the memory access time is constant for each byte of physical memory. On some systems, however, this assumption fails and access time varies with location; we refer to these systems as having a NUMA (non-uniform memory access) design.

   Consider a NUMA system where each core has access to a small amount of (relatively) fast physical memory and a larger amount of (relatively) slow physical memory. Assume that memory management hardware can map process virtual addresses to either part of physical memory. How does this design complicate the address space abstraction? At a high level, describe a way to make use of this NUMA property. (One of the systems design principles we have discussed may come in handy.)

UB ID:

6. (5 points) Describe two scheduling algorithms, only one of which can be from the "no-nothing" group. For each, provide a one sentence explanation and describe one pro or con of the approach.

Virtual Page Number	Physical Page Number	Permissions
4366	8308	RE
5437	578	RW
4758	5133	R
5173	8179	W
365	8308	RWE
94	2	R

Table 1: **Page Table.**

7. (5 points) Given the current process page table above, indicate the result of the following five load, stores, and fetches (load and execute.) **Note: to make things easier on everyone the question uses base-10 arithmetic and 1000-byte pages.**

   1. store `365004`
   2. fetch `475876`
   3. load `94230`
   4. fetch `234900`
   5. store `4366100`

UB ID:

Long Answer

Choose 1 of the following 2 questions to answer. **Do not answer both questions.** If you do, we will only read one, most likely the one that looks shorter and more incorrect. If you need additional space, continue and clearly label your answer on other exam sheets.

8. (20 points) Choose one of the following two questions to answer:

   1. **Interactivity Detection on Smartphones.** Smartphones are the fastest growing computing platform, with most now running platforms built on top of operating systems with roots in desktop computers such as Linux (Android) and Windows. Just as we discussed in class, performing effective thread scheduling on mobile devices can have a big impact on performance. Mobile smartphones, however, have some important differences from older computing devices that impact the scheduling process, particularly when considering *interactivity*.

      First, describe the interactivity detection problem and explain why this information is generally important to thread scheduling. Continue by presenting two reasons why interactivity detection could be even more important on mobile smartphones. You will want to consider patterns of use, changes in the environment caused by mobility, and constraints specific to smartphones.

      Second, consider how the differences between smartphones and traditional devices affect the interactivity detection problem and propose a new approach to interactivity detection that responds to these differences. You will want to think about what is different about how users interact with mobile phones, as well as the different features on these devices compared to laptops and desktops.

---

   2. **Jumbo Pages.** While operating system pages have traditionally been 4K, some modern operating systems support "jumbo" pages as large as 64K. Based on your excellent and fast implementation of virtual memory for CSE 421 ASST3 you are hired as a kernel developer for the new Lindows © operating system company. Unfortunately, your boss didn't take CSE 421 and derives most of his understanding of operating systems from the movie "Her"[1]. At present, Lindows does not support jumbo pages, but once your boss hears about them he is desperate to include them into Lindows © version 0.0.0.2. He asks you for help.

      First, explain why how and in what cases 64K pages would improve or degrade OS performance. What information about virtual memory use could help the OS decide whether to locate content on a jumbo or regular-sized page? Second, explain how, in certain cases, you can implement jumbo-page-like functionality on top of an existing system that supports 4K pages *without* changing the underlying memory management hardware. What MMU features are required for this to work? Which benefits of jumbo pages are preserved or lost by your approach?

[1]Her is a 2013 American science fiction romantic comedy-drama film centering on a man who develops a relationship with an intelligent OS with a female voice and personality. (Wikipedia)

Scratch. **Please indicate what question you are answering.**

UB ID:

Scratch. **Please indicate what question you are answering.**

UB ID:

Scratch. **Please indicate what question you are answering.**

UB ID:

BLANK