Operating Systems cs5460/6460 **Name (Print):** _____
Spring 2014
Midterm
06/03/14
Time Limit: 9:10am - 10:20am

- **Don't forget to write your name on this exam.**
- **This is an open book, open Internet exam. But no online or in-class chatting.**
- **Ask me if something is not clear in the guestions.**
- **Organize your work**, in a reasonably neat and coherent way, in the space provided. Work scattered all over the page without a clear ordering will receive very little credit.
- **Mysterious or unsupported answers will not receive full credit**. A correct answer, unsupported by explanation will receive no credit; an incorrect answer supported by substantially correct explanations might still receive partial credit.
- If you need more space, use the back of the pages; clearly indicate when you have done this.

Problem	Points	Score
1	15	
2	35	
3	40	
Total:	90	

1. Basic page tables.

   (a) (5 points) If you want to implement page tables as a one-level mechanism, i.e., as an array, how much space do you need for a page table that covers a 4GB address space? How does it compare to a 2-level page table used in the x86 architecture.

   (b) (10 points) Suggest a design of a page table mechanism that supports 1KB pages. Draw a picture, and provide some discussion. What are the advantages of 1KB pages? What are the disadvantages?

2. Below is a code of xv6 page allocator. To test how well this allocator scales with the number of CPUs, you wrote a synthetic benchmark that allocates and deallocates pages in loop. You run this benchmark on machines that are configured to have 1, 2, 4, 8, and 16 CPUs. For each CPU configuration you start a benchmark process on each CPU. You notice, that performance of the allocator—the number of allocation operations per second—does not go up significantly beyond performance achieved on a 4 CPU machine.

```
2834 // Allocate one 4096byte page of physical memory.
2835 // Returns a pointer that the kernel can use.
2836 // Returns 0 if the memory cannot be allocated.
2837 char*
2838 kalloc(void)
2839 {
2840     struct run *r;
2841
2842     if(kmem.use_lock)
2843         acquire(&kmem.lock);
2844     r = kmem.freelist;
2845     if(r)
2846        kmem.freelist = r>next;
2847     if(kmem.use_lock)
2848        release(&kmem.lock);
2849     return (char*)r;
2850 }
```

(a) (5 points) Why performance of the allocator doesn't go up with the number of CPUs?

(b) (10 points) You decide to optimize the allocator by using various scalability techniques. Your first attempt is to use the read-copy-update mechanism. Justify whether the RCU approach does or doesn't make sense. You can provide code for the RCU allocator if it helps.

(c) (10 points) You are not giving up. Now you decide to try hardware transactional memory. Lets assume you have to primitives: 1) `xbegin()`—starts the transaction; 2) `xend(label)`—ends the transaction. The label which is passed to the `xend()` as an argument is a label used by the "goto" mechanism in C. `xend()` jumps to this label if transaction aborts to do a restart. Modify the allocator to use `xbegin()` and `xend()`. Write the code. Do you think performance goes up? Justify your answer.

(d) (10 points) You're really eager to make it fast. Suggest a technique. Provide a high-level discussion.

3. You are given a task of porting xv6 on the hardware that is identical to x86, but does not have a paging mechanism.

   (a) (10 points) How will you implement address spaces? Remember that address spaces provide two key properties: illusion of a private memory, and isolation. Draw a figure of an address space layout for 2 processes and the kernel. Provide discussion of the mechanisms involved into your implementation.

   (b) (10 points) Remember that user processes on xv6 have only one interface to change their memory allocation—the `sbrk(n)` system call that allows the process to change its memory allocation growing it by n bytes (or shrinking it if a negative value is provided). How will you support `sbrk()` in your xv6 port? What are the data structures required? Provide a design discussion.

(c) (10 points) What if two processes want to share a region of memory? Can you suggest an interface and implementation for your port? What are the limitations of this mechanism, e.g. how many processes can share a region of memory simultaneously, how many sharing regions can be established?

(d) (10 points) Discuss advantages and disadvantages of giving up the paging mechanism.