CS 4760	**Operating Systems**	Test 1
Name:	Spring 2014	Max Pts: 50

Important: This is an open book test. You can use any books, notes, or paper but no electronic device. *Do not log into the computer during the test, or use any electronic or communications device. Change your cell phones to silent mode.* Any calculations and rough work can be done on the back side of the test pages. If there is a syntax error in any program segment, just write it down and you will get full credit for the problem. You will lose five points for not writing your name.

1. [10 pt] A machine uses a simulation of DMA to transfer data between CPU and disk drive. However, the CPU controls all the transfers and is held up while the data transfer takes place. The CPU has been benchmarked to work at 3GHz, with an average instruction requiring 1.5 clock cycles. The disk rotation speed is 7200 RPM. If our code has to perform one data transfer every 10 seconds, what is the percentage degradation in CPU performance? Ignore the actual transfer time from computation.

   Number of cycles/sec $= 3 \times 10^9$
   Number of instructions/sec $= \frac{3\times10^9}{1.5} = 2 \times 10^9$
   Number of instructions in 10 sec $= 2 \times 10^{10}$

   Time for 1 rotation of disk $= \frac{60}{7200} = \frac{1}{120}$ sec
   Since the average latency of disk will be half rotation,
   Time for $\frac{1}{2}$ rotation of disk $= \frac{1}{240}$ sec

   Number of instructions lost during half rotation $= \frac{1}{240} \times 2 \times 10^9 = \frac{1}{12} \times 10^8$

   Percentage instructions missed in 10 sec

   $$\begin{aligned} &= \frac{\frac{1}{12}\times10^8}{2\times10^{10}} \times 100 \\ &= \frac{1}{2400} \times 100 \\ &= \frac{1}{24} \end{aligned}$$

2. [6 pt] How can you ensure that the machine language instructions are executed atomically?

   In a typical machine architecture, CPU works in three stages: fetch, execute, and interrupt handling. The interrupts are handled only after the instruction is executed. Thus, the staged handlng of interrupts ensures that the instructions are executed atomocally.

3. [6 pt] What are IRQs? What is the difference between a *short* and a *long* IRQ?

   IRQs are interrupt requests. In Linux, short IRQs are expected to take a short period of time to execute during which the rest of the machine is blocked and no other interrupts are handled. Long IRQs can take longer and allow other interrupts to take place.

4. [12 pt] Show that the bakery algorithm solves the critical section problem correctly. Comment on the fact that it allows two processes to choose the same `number`.

   Mutual exclusion. A process examines the number allocated to all the other processes and proceeds only if no other process has a number smaller than the examining process. If process j picks a number after process i has examined it, it is guaranteed to get a number larger than that of process i.

   Progress. If a process has not picked a number, the examining process does not look at it any more. Thus, such process cannot stop the examining process from entering critical section.

   Bounded wait. A new process will not be able to bypass a waiting process because it will always pick a number larger than the waiting process.

   Two processes may choose the same number and that issue is resolved by the ordered pair comparison. If two processes have the same number, the process with the smaller PID gets into the cirtical section first.

5. [6 pt] How do you achieve the effect described by the following statement: "Program changes but process remains." Specify a system call used to achieve this.

   Program is only a small part of the process. Program constitutes the code and data segments of the process and may be replaced by any system call from the exec family of system calls.

6. [10 pt] Assume that the machine you are working on does not have a `test_and_set` instruction. However, it does have an indivisible `swap` instruction. This indivisible `swap` instruction swaps the contents of two memory locations atomically. How can you use this to solve the critical section problem. Show with the template for the critical section problem.

```
extern int lock = 0;           // Lock in shared memory initialized to 0

void process ( const int i )   // Code for ith process
{
    do
    {
        // Entry section

        int flag = 1;
        do
            swap ( flag, lock );
        while ( flag == 1 );

        critical_section();

        // Exit section

        swap ( flag, lock );

        remainder_section();
    } while ( 1 );
}
```