Name:

Graduate Operating Systems
Spring 2015
Calibration Exam

Closed books, notes, cell phones, PDAs, iPods, laptops, etc. No headphones, please.

You have 30 minutes to solve as many problems as you can. The problems are from undergraduate OS tests in Fall 2014. For the purpose of this test, each problem is worth 10 points.

1. **Short attention span**:

   a. A _____ is a control structure that contains the key information needed by a Unix operating system for a particular file.

   b. The _____ shows the frame location for each page of the process.

   c. _____ in a computer system is organized as a linear, or one-dimensional, address space, consisting of a sequence of bytes or words.

   d. _____ is the range of memory addresses available to a process.

   e. _____ is transparent to the programmer and eliminates external fragmentation providing efficient use of main memory.

   f. The _____ page replacement policy results in the fewest number of page faults.

   g. _____ is a storage allocation scheme in which secondary storage can be addressed as though it were part of main memory.

   h. When the system spends most of its time swapping pieces rather than executing instructions it leads to a condition known as _____ .

i. The scheduling strategy where each process in the queue is given a certain amount of time, in turn, to execute and then returned to the queue, unless blocked, is referred to as:
   1. Prioritization
   2. Round-Robin
   3. LIFO
   4. All of the above

j. The processor execution mode that user programs typically execute in is referred to as:
   1. User mode
   2. System mode
   3. Kernel mode
   4. None of the above

2. True or False?

a. In a multiprogramming system the main memory is not generally occupied simultaneously by multiple processes.

b. A hardware mechanism is needed for translating virtual addresses to physical main memory addresses at the time of execution of the instruction that contains the reference.

c. A physical address is the location of a word relative to the beginning of the program and the processor translates that into a virtual address.

d. All segments of all processes must be of the same length.

e. It is impossible to have both paging and segmentation in the same system.

3. **Short attention span, again:** Answer the following questions:

a. What is the difference between a virtual address and a physical address?

b. Which of the following instructions should be privileged?
   i. Read the clock
   ii. Set value of timer
   iii. Turn off interrupts
   iv. Switch from user to kernel mode
   v. Clear memory

c. Give an example of a preemptive scheduling algorithm.

d. True or False: A memory system that uses paging is vulnerable to external fragmentation. Why or why not?

e. Of the following items, which are stored in the process control block?
   i. Page table pointer
   ii. Page table
   iii. Stack pointer
   iv. Segment table
   v. List of processes in the ready state
   vi. The content of CPU registers
   vii. Program counter

4. Consider the following program executed by two different processes P1 and P2:

```
{
      shared int x;                          //s1
      x = 10;                                //s2
      while (1) {
            x = x - 1;                       //s3
            x = x + 1;                       //s4
            if (x != 10)                     //s5
                  printf("x is %d",x)        //s6
      }
}
```

Consider that the processes P1 and P2 are executed on a uniprocessor system. Note that the scheduler in a uniprocessor system would implement pseudoparallel execution of these two concurrent processes by interleaving their instructions, without restriction on the order of the interleaving.

a. Show a sequence (i.e., trace the sequence of interleavings of statements) such that the statement "x is 10" is printed. Suggested format:
   Pi: <instruction> <relevant new value >
b. Show a sequence that leads to the statement "x is 8" be printed.

Hint: Remember that the increment/decrements at the source language level are not done atomically, e.g., the assembly language code below implements the single C increment instruction (x = x + 1).

```
LD R0,X /* load R0 from memory location x */
INCR R0 /* increment R0 */
STO R0,X /* store the incremented value back in X */
```

5. **(10 points)** You are given a bunch of scientific code that contains loops of the form:

```
for (i = 0; i < n; i++)
     for(j = 0; j < n; j++)
          a[j][i] += b[j][i];
```

The matrices `a` and `b` are allocated using code such as:

```
a = malloc (n* sizeof(*a));
for (i = 0; i < n; i++)
     a[i] = malloc(n* sizeof(a[i]));
```

While running the program, you notice that the addition loop trashes the TLB much more than it should. Say why, and give a simple fix. In general, what do you expect to happen to paging performance?

6. Write the reference string generated by the following program, considering the following parameters: the system has 256-byte pages. The program is located at address 1020, and its stack pointer is at 8190 (the stack grows toward 0). Each instruction occupies 4 bytes (1 word), and both instruction and data references count in the reference string.

```
Load word from address 6144 into register 0
Push register 0 onto the stack
Call procedure from address 5120, stack the return address
Subtract the immediate constant 16 from the stack pointer
and place it in register RC
Compare the value in RC to the immediate constant 4
Jump if equal to 5152
```

Reference String:

7. Sketch rough graphs ("back of napkin") illustrating the relationships among the following quantities. Note any important assumptions underlying your analysis.
    a. Page fault rate as a function of page size. What page sizes are typical for current processors and operating systems?
    b. Disk access latency as a function of rotation speed.
    c. CPU utilization as a function of the degree of multiprogramming. Consider a uni-processor system.
    d. Memory fragmentation as a function of page size. Is that internal or external?