CS162 Fall 2004 **Professor Alan Jay Smith**

Midterm 1, Monday, October 11, 2004

(Total Points Possible: 100)

1. What's an inverted page table? What are the relationships between the size of the inverted page table, the size of the virtual address space, and the size of the physical address space? (14)

2. Suppose that you have an architecture with a base register (for relocation) but no bounds register. What is the advantage of this design relative to a system with no base register? What is the disadvantage of this design relative to a system with both base and bounds registers? (14)

3. Assume a system with segmentation but no paging. Assume that the system uses a segment table. Please show how a virtual address is used to make a memory reference. (Be sure to show all segment table entry fields and their uses, any necessary registers, comparators, adders, etc.)
   (15)

4. For the following two cases, please either show a complete safe sequence or prove that there isn't one:
(12)

Process	has-X	has-Y	max-needs-X	max-needs-Y
A	35	10	65	40
B	55	80	105	220
C	35	20	110	50
D	0	70	50	90

(a) available: X: 40 Y: 40
(b) available: X: 40 Y: 35

5. Prove that Shortest Job First is optimal (minimum average time to completion, averaged over all of the jobs) in a system which starts with exactly N jobs, each of whose run times are known, and no which no further jobs arrive.
(16)

6. Suppose that you had to schedule jobs for a printer. Assume that the system uses spooling - i.e. print files are written to disk and then are printed without interruption after they are complete.
(16)

   (a) What's the optimal (realizable) algorithm that minimizes flow time, and why?
   (b) Now suppose that you have 2 printers, and suppose that print job sizes are highly skewed. In order to provide good service to your users, how might you want to modify the scheduling algorithm you selected for part a? (Keep in mind the "real goal" of scheduling algorithms.) Please explain and justify your answer.

7. Assume that we have atomic operations: [increment value in memory, and then load the result] and [decrement value in memory].

   The following "code" was given in lecture to obtain mutual exclusion for a critical section. What's wrong with it? Please explain.
   (13)

```
Init: A = 0

loop: {increment A in memory, load A},
if A~=1, then {decrement A in memory}, go to loop

critical section here

{decrement memory location A}
```