Exam 2 - Official Solutions (i.e. prepared by the professor and the TAs)

Graded by Smith:

1. Assume that the mean job processing time is 10, and that the task switching overhead is 0.5. (I.e. at the end of every interval of length Q, the CPU experiences a task switch event.) Assume round robin scheduling. Assume that Q (the round robin quantum) ranges from .001 to 1000. Plot the shape of the mean flow time as a function of Q and explain. (No credit unless your explanation matches your plot.) (9)

The situation is just the same as in assignment 1: we have a computer system (with job run times that are highly skewed), and round robin scheduling.

When the quantum is much less than the mean job processing time (job run time), then there is a huge level of overhead due to task switching. (For Q=.001, overhead is 99.8%).

As the quantum increases, mean flow time (time from arrival at the rear of the queue until completion and departure) decreases, until the quantum is in the neighborhood of the mean run time. (Not necessarily equal to the mean run time.) As the quantum continues to increase, the mean flow time increases again (but to a much lesser extent than for very small quanta), since as Q--> infinity, RR-->FIFO, and FIFO has much higher flow time than RR for highly skewed job run times.

Notes:

1. a couple of people described job run times as "exponential". They are NOT. Expected time to completion for exponential distribution is constant.
2. some people misread the question and thought that a job continued to use the processor through the end of the quantum, even if it didn't need it. I gave some partial credit for that.

---

2. Suppose we have a disk with 512 cylinders, and the disk is currently at cylinder 110 (and has previously just processed a request for cylinder 105) and the disk queue contains read/write requests for sectors on cylinders 84, 302, 103, 96, 407 and 113. (15)

2a. How far must the head travel to satisfy the requests in the queue using FCFS scheduling? (If the arm goes from cylinder 10 to 20 to 14, that is a total of 16 cylinders moved.)

2b. How far must the head travel to satisfy the requests in the queue using the SCAN arm scheduling strategy?

2c. How far must the head travel to satisfy the requests in the queue using CSCAN scheduling?

2d. How far must the head travel to satisfy the requests in the queue using the SSTF arm scheduling strategy?

2e. Assuming no further arrivals, what is the optimal order (i.e. sequence of cylinder numbers) in which to service the requests, and how far does the head travel?

SSTF:
110-113-103-96-84-302-407 355 cylinders

FIFO (FCFS)
110-84-302-103-96-407-113 1055 cylinders

SCAN
110-113-302-407-103-96-84 620 cylinders
(2 points for 110-113-302-407-512-103-96-84 = 830 cyl)

CSCAN
110-113-302-407-84-96-103 639 cylinders
(2 points for 110-113-302-407-512-1-84-96-103 = 1017 cyl;
no credit for 110-113-302-407-512-84-96-103)

Optimal
110-84-96-103-113-302-407 349 cylinders

Notes:
My original plan was to give no partial credit, which is why I didn't ask you to show your work. There were two common errors, however, that seemed to deserve partial credit.
I gave 2 points (half credit) for an answer IF there was a simple arithmetic mistake and the sequence of cylinders was shown.
I gave no credit if the answer was wrong, and it wasn't obvious that you knew what you were doing. I didn't try to debug your answer.
Neither SCAN nor CSCAN continues to the edge of the disk if there are no requests to service in that direction. You got half credit if

you assumed that (and computed it correctly and consistently).
If you didn't do the arithmetic, you didn't get credit. Addition of a column of 3-digit numbers should have been mastered in the third grade.
If you made 2 or more errors on a part of the question, you got no credit for that part.

Graded by Mukund Seshadri:

3. Describe and explain at least two reasons why disabling interrupts to implement synchronization primitives is not sufficient to provide mutual exclusion. Is disabling interrupts necessary? Is it useful? Explain your answers. (12)

(a)Two reasons why disabling interrupts to implement synchronization primitives is not sufficient to provide Mutual exclusion:

(1) Would not work for a multiprocessor environment, since disabling ints on 1 processor does not mean that interrupts are disabled on all the processors having access to shared data.
(2)Some interrupts (e.g. traps) cannot be disabled.

(b)Disabling interrupts is not necessary - other methods can be used - for example, if there is hardware support for the Test-and-Set instruction, we can use a method based on that.

(c) Disabling Interrupts is useful for efficiency - if a context switch occurs in the Critical Section in the synch. primitive implemenation, other processes can spin idly (in the spi-lock implementation) - so we want to avoid context switches as far as possible during the CS phase of the implementation of the Synch. primitive.

6a. What is a stack algorithm? What is the advantage of a paging algorithm being a stack algorithm?

(a)A stack algorithm is one that satisfies the inclusion property. The inclusion property states that, at a given time, the contents(pages) of a memory of size k page-frames is a subset of the contents of memory of size k+1 page-frames, for the same sequence of accesses. The advantage is that running the same algorithm with more pages(.ie. larger memory)

will never increase the number of page faults.

6b. Is LRU a stack algorithm? Prove or disprove.

(b) Yes, LRU is a stack algorithm. In LRU, the k most recently accessed pages is always a subset of the k+1 most recently accessed pages. So it satisfies the inclusion property and is hence a stack algo.

6c. Is FIFO a stack algorithm? Prove or disprove.

(c)No, FIFO is not a stack algorithm. See Q-5 FIFO for the Counter-Example.

(Reproduced below - Note the columns with the asterisks above them - they show that FIFO violates the inclusion property)

FIFO:

```
              * *     *
  7 6 5 4 7 6 8 7 6 5 4 8
+---------------------------------
|>7 7 7 7 7 7 >8 8 8 8 >4 4
|  >6 6 6 6 6 6 >7 7 7 7 >8
|    >5 5 5 5 5 5 >6 6 6 6
|      >4 4 4 4 4 4 >5 5 5
```

Page faults with 4 page frames = 10

```
              * *     *
  7 6 5 4 7 6 8 7 6 5 4 8
+---------------------------------
|>7 7 7 >4 4 4 >8 8 8 8 8 8
|  >6 6 6 >7 7 7 7 7 >5 5 5
|    >5 5 5 >6 6 6 6 6 >4 4
```

Page faults with 3 page frames = 9

Graded by Morley Mao:

5. For the following page reference string, please show the number

of page faults for a memory of size 3 page frames, and for a memory
of size 4 page frames, using LRU, FIFO and OPT replacement.
Please show your work.

7 6 5 4 7 6 8 7 6 5 4 8

	Mem size 4	3
OPT	6	7
LRU	8	10
FIFO	10	9

Note the Belady's anomaly for FIFO.

Detailed answer:

LRU(3)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m m m m h h m m m
memory content: 7 7 7 4 4 4 8 8 8 5 5 5
6 6 6 7 7 7 7 7 4 4
5 5 5 6 6 6 6 6 8

10 page faults

LRU(4)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m h h m h h m m m
memory content: 7 7 7 7 7 7 7 7 7 7 7 8
6 6 6 6 6 6 6 6 6 6
5 5 5 5 8 8 8 8 4 4
4 4 4 4 4 4 5 5 5
8 page faults

FIFO(3)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m m m m h h m m h
memory content: 7 7 7 4 4 4 8 8 8 8 8 8
6 6 6 7 7 7 7 7 5 5 5
5 5 5 6 6 6 6 6 4 4
9 page faults

FIFO(4)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m h h m m m m m m
memory content: 7 7 7 7 7 7 8 8 8 8 4 4
6 6 6 6 6 6 7 7 7 7 8
5 5 5 5 5 5 6 6 6 6
4 4 4 4 4 4 5 5 5
10 page faults

OPT(3)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m h h m h h m m h
memory content: 7 7 7 7 7 7 7 7 7 5 4 4
6 6 6 6 6 6 6 6 6 6 6
5 4 4 4 8 8 8 8 8 8
7 page faults

OPT(4)
page access: 7 6 5 4 7 6 8 7 6 5 4 8
hit or miss: m m m m h h m h h h m h
memory content: 7 7 7 7 7 7 7 7 7 7 7 7
6 6 6 6 6 6 6 6 6 6 6
5 5 5 5 5 5 5 5 5 5
4 4 4 8 8 8 8 8 8

6 page faults.

Graded by Umesh Shankar:

4a. What is a TLB and what is it used for?

a) Short answer: The TLB is a hardware-based cache for the page table. This speeds up memory accesses since no additional memory references are needed to do the translation on a cache hit.

4b. Three ways that a TLB can be designed are fully associative, set

associative and direct mapped. Explain all 3 (including the tradeoffs between them). Use diagrams.

b)
The three strategies are different ways of mapping page table entries into cache slots.

Direct mapped: the low bits of the virtual address are used to select the slot; thus, each entry may go in only one slot. This type of cache is simple to build, so cycle times can stay low; it can be made relatively large; it often has a poor miss ratio since there can be a lot of contention for slots.

Fully associative: the other end of the spectrum. An entry may go in any slot, so replacement is more flexible. All slots are searched in parallel by the hardware. This complicates the lookup mechanism, so such caches are smaller and may increase the cycle time. The miss ratio, however, tends to be very low.

Set associative: a compromise between the two. Some low bits are used to select a set into which the entry may go. Lookups search the set in parallel. The properties of this type of cache are in between those of the above two.

For an N-slot cache (TLB), we may view the direct mapped strategy as 1-way set associative and the fully associative as N-way set associative (hence the term).