CE-6
August 2017 QE

1. (20 pts, 2 pts each) For each of the statements below, indicate whether the statement is true or false. Explanations may allow partial credit for wrong answers.

   - When the binary a.out for the following program runs, the printf statement below will output value 2017.

     ```
     void main() {   int local = 2016;
        int pid = fork(); if (pid == 0) {
           execlp("/bin/a.out", "a.out", NULL);
           local = 2017;
           printf("The child process: local=%d", local);
        } else {
           wait(NULL);
           exit(0);
        }
      }
     ```

   - Assume synchronization primitives are correctly supported by the OS. If a multi-threaded user program using the synchronization primitives is known to execute correctly on a time-shared (i.e. preemptively scheduled) uniprocessor, then it will execute correctly on a multiprocessor.

   - A CPU scheduling policy that minimizes the average turn around time can not lead to starvation.

   - In a pure segmentation system, without the help of swapping in and out, or moving other segments around, it is possible to enlarge a segment as long as there is a contiguous chunk of free memory in DRAM that is at least as large as the requested enlargement.

   - In a pure one-level paging system, the page table is kept in the MMU.

- The hardware maintains consistencies between the TLB and DRAM, i.e., whenever a page table entry that is cached in TLB is changed in the physical memory, the cached copy in TLB is automatically updated.

- A binary semaphore can only be used by two processes/threads for synchronization between them.

- In the Unix File System, the name of a hardlink is stored in its inode.

- In the Unix File System, the name of a softlink is stored in its inode.

- In the Unix File System, the operation "ls -i" which lists the inode number along with each file name in the directory requires accessing more disk blocks and hence is slower than "ls".

2. (30 pts, 3 pts each) Answer each of the following questions concisely but precisely.

   - What is a system call?

   - In wait-free synchronization, for example in the singly-linked queue insertion example below, we still need to put a while loop in the atomic commit operation. So why is this called "wait-free synchronization"?

```
QElem *queue;

void Insert(item){
        QElem *new = malloc(sizeof(QElem)); new-
        >item = item;
        do {
                new->next = ldl(&queue);
        } while (!stc(&queue, new));
}
```

   - Why does a context switch between two threads in the same process have less overhead than a context switch between two threads in different processes?

   - Why do we still disable interrupts to prevent context switch in addition to using the lock with TAS in the following implementation of wait() for multiprocessors?

```
void wait(semaphore s)
{
disable interrupts;
while (TAS(s->lock, 1) == 1);
if (s->count > 0) {
  s->count--; s->lock = 0; enable interrupts;
  return;
}
add(s->q, current_thread);
s->lock = 0;
enable interrupts; sleep();
}
```

]

- Can FIFO ever be the worst possible non-preemptive CPU scheduling algorithm in terms of average turn around time? If so, under what circumstances? If not, explain why not.

- Can round robin ever be the worst possible CPU scheduling algorithm in terms of turn around time? If so, under what circumstances? If not, explain why not.

- In a paging system, what is the motivation for using an inverted page table?

- What is the challenge/downside in using an inverted page table?

- In a paging system, when malloc(8193) invoked by user process P successfully returns, how many physical pages have been allocated to process P, assuming a page size of 4KB? If the answer is more than 1, are they consecutive in the physical memory?

- Is it quicker to write 1 block in a RAID Level 5 organization with 5 disks (4 for striping data blocks and 1 for storing parity blocks) than in a RAID Level 1 (mirroring) organization with 2 disks (i.e., mirroring only)?

3. (TLB - 20 pts)

   (a) (4 pts) How does a Translation Look-aside Buffer (TLB) improve the performance of paging?

   (b) (Paging – 8 pts) Consider a two-level paging system with the page table stored in memory.

      i. (4 pts) If a memory reference takes 300 nanoseconds, how long does a paged memory reference take?

      ii. (4 pts) If we add TLBs, and 75 percent of all page-table references (i.e. the needed PTEs) are found in the TLBs, what is the effective (i.e. average) memory reference time? (Assume that finding a page-table entry in the TLBs takes zero time, if the entry is there.)

   (c) (8 pts) What happens to the TLB with a context switch of processes? (Careful: there are two very different answers here, depending on what the hardware TLB stores in each TLB entry; specify what your assumptions are.)

4. (Page Replacement – 20 pts)

(a) (4 pts) A page replacement algorithm belongs to the class stack algorithms if it can be shown that given any sequence of virtual page references, the set of pages that would be in memory with n physical pages is always a subset of the set of pages that would be in memory with n+1 physical pages. Which of the following page replacement algorithms are stack algorithms? (1) OPT (2) LRU (3) FIFO (4) LIFO (5) LFU (Least Frequently Used)

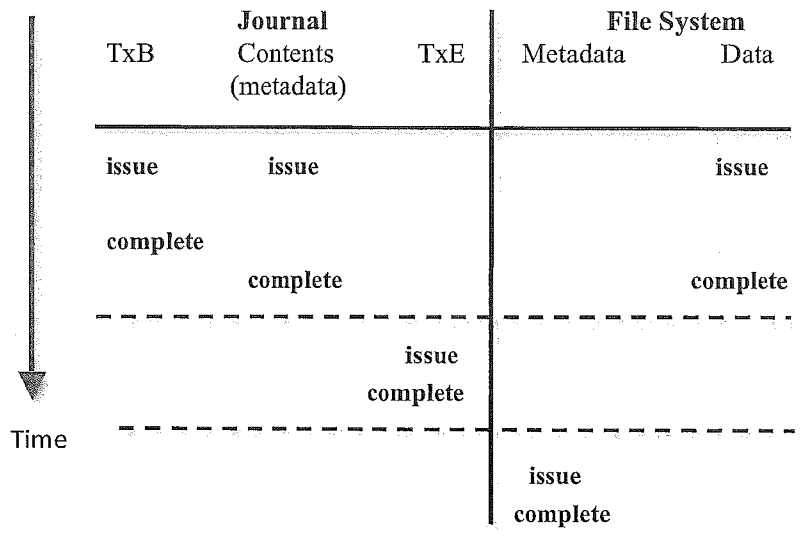
(b) (8 pts) Consider the following virtual page reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, 1, 2. How many page faults will there be under the FIFO replacement algorithm on Tom's PC which has 4 physical pages? How many page faults will there be on Jerry's machine which has 3 physical pages? Assume the physical memory is initially empty.

(c) (8 pts) Can you prove OPT is a stack algorithm?

5. (Journaling File System - 10 pts) In data journaling (as in Linux ext3 file system), the file system journals all user data in addition to metadata. A simpler form of journaling, called metadata journaling, only journals metadata, and user data is not written to the journal.

   (1) (5 pts) What is the motivation for metadata journaling, in contrast to data journaling?

   (2) (5 pts) Why in metadata journaling, do we need to order the user data block write and journal commit, i.e., why do we have to wait until writing the user data block to the file system is complete before committing the journal (writing the transaction commit block containing TxE to the journal), as shown in the following timeline?



Write in Exam Book Only