Final Exam for CSE 153 (Fall 2017)

15th *December 2017* **Name:**

Student ID:

Instructions:

***** This exam is out of a total of 20 points.

***** Be brief in your answers. You will be graded for correctness, not on the length of your answers.

* Make sure to write legibly. Incomprehensible writing will be assumed to be incorrect.

* Read all questions carefully. Every word is in there for a reason.

I. Indicate whether any 10 of the following statements are true or false (**5 points**)

___F____ Segmentation can lead to internal fragmentation

___F____ Swap can be smaller than physical memory

___T____ Swap resides on disk

___T____ Page tables can be bigger than physical memory

____F___ Page faults are handled by hardware

____F___ Journaling FS improve the performance of FFS

____T___ Page replacement can require invalidating a TLB entry

____F___ Multi-level page tables improve the performance of address translation

____F___ The working set of a process includes all its data pages

____T___ free () does not need a system call

____T___ Explicit free list is better than implicit free list for heap management

II. (**5 points**) Consider a memory system with a virtual address space of 32 bits, a physical address space of size 30 bits, and a page size of 4Kbytes.

a. (1 point) Given the virtual address 0xff0beef, what is the value of the VPN and the offset?

VPN: 0xff0b

Offset: 0xeef

b. (1 point) Given two pointers (i.e., virtual addresses) VA and VB with values 0xff00423 and 0xff01321 respectively, which of the following is true about their physical addresses. Assume that both pages are in memory.

   a. VA's physical address is larger than VB's physical address
   b. VB's physical address is larger than VA's physical address
   c. It depends on whether we have a TLB hit or miss
   d. Its impossible to tell

c. (1 point) Consider now the following two addresses PA and PB with values 0xff00423 and 0xff00321 respectively.

   a. VA's physical address is larger than VB's physical address
   b. VB's physical address is larger than VA's physical address
   c. It depends on whether we have a TLB hit or miss
   d. Its impossible to tell

d. (1 point) Can the virtual address of a global array change over the lifetime of a program (without the program copying/moving it)? Can the physical address?

VA of a data structure wont change unless you move it. PA can change after a page moves to swap then back in.

e. (1 point) Someone argues that the clock algorithm is supposed to be an approximation of LRU – Take a position for or against this statement, and briefly justify it.

The reference bit being accessed allows a page not to be replaced for a full round. So, the page that is being replaced has not been accessed for a full round. Note that this is approximately LRU in the sense that removed page was not accessed recently but it may not be the one that was least recently used. Also, in situations where most pages are getting referenced, clock becomes almost FIFO.

III. (**5 points**) In the following problem, consider a unix file system with i-nodes similar to the one we discussed in class. It has 8 direct links, 1 indirect link, 1 double indirect link and 1 triple indirect link. The block size is 1 Kbyte. Assume a pointer is 4 bytes in size.

(a) (1 point) What is the maximum file size that does not use the triple indirect link in inode?

8 directly indexed blocks + 256 single indirect indexed blocks + 256*256 doubly indirect indexed blocks.

(b) (1 point) Suppose that you write one more block worth of data at the end of the file in part (a), how many additional blocks will be needed and the new structure of the file

Need to add the data block + top level of triple indirect, one block of the next level of triple indirect, and one block of the bottom level. 4 blocks total. So, in addition to the 8 direct blocks, single indirect block, and the double indirect block all filled out (from part a), we will have the triple indirect->top level of indirect with one pointer->next level index block with one pointer->bottom level index block with one pointer->new data block.

(c) (1 point) How many blocks total on the disk will have to be updated to carry out the operation in part (b). Note that some other blocks not related directly to the file may have to be updated.

Free map to update the 4 newly allocated blocks, 4 writes to write the new blocks (3 index one data), one write to update the i-node to put the new triple index pointer.

(d) (2 points) Your computer crashes in the middle of the operations above; explain what could cause the disk to be inconsistent? Show a specific scenario with the operations in part c and explain what is the observed effect of the inconsistency.

Caching and disk scheduling may cause delay of some or all of the writes, or reordering them leading to scenarios where any subset of the writes could have happened, with the others lost. Many scenarios possible, for example, data block written only, but not connected to in-node (data lost). Free map not updated (blocks will get reassigned to other files). Free map updated but data block not written (block lost/unsable), and so on…

IV (5 + 1 bonus point) Consider a byte-addressable system with 1 Gbyte physical memory that uses a 2-level page table. The directory (i.e., outer page table) has a size of 16Kbytes. The PTE size is 8 bytes. Its ok to leave expressions if they are too difficult to simplify.

(a) (1 point) What is the size of the inner page table ***in pages***? (Hint: you can infer this from the directory size)

Outer page table is 16Kbytes, which is 16K/8 PTEs or 2K PTEs. The inner page table has 2K pages.

(b) (1 points) If the page size is p bytes, what is the number of PTEs in the inner page table as a function of p (Hint: use the result you derived in part (a)).

There are 2K pages from (a) each of size p bytes, so the total is 2K*p bytes. This is a page table, so the number of PTEs is the size/size of PTE = 2048*p/8 = 256p

(c) (1.5 points) Given also that the Virtual addresses are 36 bits wide, write another expression for the number of PTEs in the inner page table as a function of the page size, p.

2^36/p is the number of pages in the address space. This is also the number of PTEs in the page table.

(d) (.5 and 1 point bonus) Given that the expressions in (b) and (c) are both for the the number of PTEs in the inner page table, solve for p to determine the page size in the system

2^36/p = 256 p.

p^2 = 2^28, p = 2^14

(e) (1 point) How many bits are there in the Physical Page number?

Since the page size is 2^14, there are 14 offset bits, leaving 16 bits for PPN.