UNIVERSITY OF TORONTO

FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION , Apr., 2014
Third Year – Materials
ECE344H1 - Operating Systems
Calculator Type: 2
Exam Type: A
Examiner – D. Yuan

Please Read All Questions Carefully! Please keep your answers as concise as possible. You do not need to fill the whole space provided for answers.

There are __19__ total numbered pages, __10__ Questions.
You have 2.5 hours. Budget your time carefully!

Please put your FULL NAME, UTORid, Student ID on THIS page only.

Name: ____

UTORid: ____

Student ID: ____

Grading Page

	Total Marks	Marks Received
Question 1	10	
Question 2	6	
Question 3	7	
Question 4	15	
Question 5	12	
Question 6	8	
Question 7	8	
Question 8	12	
Question 9	14	
Question 10	8	
Total	100	

Question 1 (10 marks, 2 marks each): True or False *(No explanations needed)*

(a) As the size of the virtual address space grows, the amount of space occupied by a single-level page table also grows.

☐True ☐False

(b) As the amount of addressable physical memory grows, the size of each page-table entry (PTE) ***must*** grow as well.

☐True ☐False

(c) In a 2-level page table, each ***master*** page table entry should have a valid bit.

☐True ☐False

(d) One of the goals for disk scheduling is to minimize the disk's total time spent on seek.

☐True ☐False

(e) In Unix File System, The metadata information for each file is stored in its i-node.

☐True ☐False

Question 2 (6 marks): Hard disk

(a)(3 marks) Assume a simple disk that has only a single track, and a simple First Come First Serve (FCFS) scheduling policy. The rotational delay on this disk is R; there is no seek cost (only one track!), and transfer time is so fast we just consider it to be free. What is the (approximate) worst case execution time for three (3) read requests (to different blocks)?

(b)(3 marks) Now assume the disk has three tracks. The time to seek between two adjacent tracks is S; it takes 2S to seek across two tracks (e.g., from the outer to the inner track). The rotational delay and transfer time are the same as above. Given a FCFS scheduler, what is the worst-case time for three read requests to different blocks?

Question 3 (7 marks): Paging

Assume now you're writing a new OS. Instead of using a page size of 4KB (2^{12} Bytes), you decide to change the page size to 4MB (2^{22} Bytes). The virtual memory address in your new OS is 32 bits.

(a)(1 mark) Assume you use 1-level page table, describe how you break the 32 bits into page number and offset.

(b)(1 mark) How many entries are there in this 1-level page table?

(c)(2 marks) Comparing to the smaller page size (4KB), what are the advantages and disadvantages of 4MB page size? List two advantages and one disadvantage.

(d)(1 marks) What is the difference between software managed TLB (e.g., MIPS) and hardware managed TLB (e.g., x86)?

(f)(2 marks) Do you need to change the hardware to support your new OS? Why? Discuss the two cases separately: software managed TLB (e.g., MIPS) and hardware managed TLB (e.g., x86).

Question 4 (15 marks): Scheduling

Scheduling policies can be easily depicted with some graphs. For example, let's say we run job A for 10 time units, and then run job B for 10 time units. Our graph of this policy might look like this:

```
    |          |
CPU |AAAAAAAAAA|BBBBBBBBBB
    |__________|__________
    0          10        20
```

In this question, you will show your understanding of scheduling by drawing a few of these pictures.

(a)(3 points) Draw a picture of Non-preemptive Shortest Job First (SJF) scheduling with three jobs, A, B, and C, with run times of 6, 3, and 2 time units, respectively. Assume they arrive at the same time.
Make sure to LABEL the x-axis appropriately.

```
    |
CPU |
    |____________________________________________________________
```

(b)(3 points) What is the average TURNAROUND TIME for jobs A, B, and C?

(c)(3 points) Draw a picture of preemptive ROUND-ROBIN scheduling for jobs A, B, and C, which each run for 6 time units. Assume a 2-time-unit time slice, and that context switch time (S) takes 1 time unit.
Make sure to LABEL the x-axis appropriately.

```
    |
CPU |
    |____________________________________________________________
```

Now assume you have a multi-level feedback queue (MLFQ) scheduler. Recall that an MLFQ scheduler groups jobs into different priorities, and for jobs in the same priority a round-robin scheduling is used.

In your MLFQ scheduler, there are 3 levels of priorities (highest priority is 2, lowest priority is 0). Assume the time-slice for the jobs with the highest priority is 1, then 2 at the middle, and 3 for the lowest priority.

(d)(3 points) Why we want to assign smaller time-slice for higher-priority jobs?

(e)(3 points) Assume two jobs (A and B), both BATCH jobs (no I/O), each with a run-time of 7 time-units, and both entering the system at T=0, priority 1. Assume the time-slice is 2. (Assume the larger the priority value is, the higher the priority.)

Draw a picture of how the scheduler behaves for these jobs. Make sure to LABEL the x-axis. *Explain your answer.*

```
           2 |
             |
  Priority 1 |
             |
           0 |_____________________________________________________
```

Question 5 (12 marks): Unix File System

Consider a very simple Unix-style file system. The INITIAL STATE, or state(i), of this FS is shown:

```
Inode Bitmap : 10000000
Inode Table : [size=1,ptr=0,type=d] [] [] [] [] [] [] []
Data Bitmap : 10000000
Data : [("." 0), (".." 0)] [] [] [] [] [] [] []
```

There are only eight inodes and eight data blocks; each of these is managed by a corresponding bitmap (a ‘1’ indicates the inode or data block is used, a ‘0’ indicates it’s free). The inode table shows the contents of each of eight inodes, with an individual inode enclosed between square brackets; in the initial state above, only inode 0 is in use. When an inode is used, its size and pointer field (ptr) are updated accordingly (in this question, files can only be one block in size; hence a single inode pointer); when an inode is free, it is marked with a pair of empty brackets like these “[]”. Note there are only two file types: directories (type=d) and regular files (type=r). Data blocks are either “in use” and filled with something, or “free” and marked accordingly with “[]”. Directory contents are shown in data blocks as comma-separated lists of tuples like: (“name”, inode number). The root inode number is zero.

Your task: figure out what file system operation(s) must have taken place in order to transition the file system from some INITIAL STATE to some FINAL STATE. You can describe the operations with words (e.g., file “/x” was created, file “/y” was written to, etc.) or with the actual system calls (e.g., create(), write(), etc.).

Note that these questions build as they go, i.e., the file system starts in the initial state “state(i)” (as above), and then goes through states(a), (b), (c), (d), (e), and finally (f) in the questions below. Thus, your task is to figure out what operation(s) caused these changes as the question progresses.

(a)(2 marks) INITIAL STATE: state(i) as above to FINAL STATE (a):

```
Inode Bitmap : 11000000
Inode Table : [size=1,ptr=0,type=d] [size=0,ptr=-,type=r] [] [] [] [] []
[]
Data Bitmap : 10000000
Data : [("." 0),(".." 0),("f" 1)] [] [] [] [] [] [] []
```

Operation that caused this change?

(b)(2 marks) INITIAL STATE: state(a) as in part (a) of this question to FINAL STATE (b):

```
Inode Bitmap : 11000000
Inode Table : [size=1,ptr=0,type=d] [size=1,ptr=7,type=r] [] [] [] [] []
[]
Data Bitmap : 10000001
Data : [("." 0),(".." 0),("f" 1)] [] [] [] [] [] [] [SOMEDATA]
```

Operation that caused this change?

(c)(2 marks) INITIAL STATE: state(b) to FINAL STATE (c):

```
Inode Bitmap : 11000000
Inode Table : [size=1,ptr=0,type=d] [size=1,ptr=7,type=r] [] [] [] [] []
[]
Data Bitmap : 10000001
Data : [("." 0),(".." 0),("f" 1),("b" 1)] [] [] [] [] [] [] [SOMEDATA]
```

Operation that caused this change?

(d)(2 marks) INITIAL STATE: state(c) to FINAL STATE (d):

```
Inode Bitmap : 11000000
Inode Table : [size=1,ptr=0,type=d] [size=1,ptr=7,type=r] [] [] [] [] []
[]
Data Bitmap : 10000001
Data : [("." 0),(".." 0),("b" 1)] [] [] [] [] [] [] [SOMEDATA]
```

Operation that caused this change?

(e)(2 marks) INITIAL STATE: state(d) to FINAL STATE (e):

```
Inode Bitmap : 11000000
Inode Table : [size=1,ptr=0,type=d] [size=1,ptr=7,type=r] [] [] [] [] []
[]
Data Bitmap : 10000001
Data : [("." 0),(".." 0),("b" 1)] [] [] [] [] [] [] [OTHERDATA]
```

Operation that caused this change?

(f)(2 marks)INITIAL STATE: state(e) to FINAL STATE (f):

```
Inode Bitmap : 10000000
Inode Table : [size=1,ptr=0,type=d] [] [] [] [] [] [] []
Data Bitmap : 10000000
Data : [("." 0),(".." 0)] [] [] [] [] [] [] []
```

Operation that caused this change?

Question 6 (8 marks): Synchronization I

You are given the following code, which adds two vectors together, and does so in a multithread-safe way.

```
void vector_add(vector *v1, vector *v2) {
  lock(v1->lock);
  lock(v2->lock);
  for (i = 0; i < v1->size; i++) {
    v1[i] = v1[i] + v2[i];
  }
  unlock(v1->lock);
  unlock(v2->lock);
}
```

Two different threads, 1 and 2, call this code as follows:

```
Thread 1:                        Thread 2:
vector_add(&vectorA, &vectorB); vector_add(&vectorB, &vectorA);
```

These two threads can be executed concurrently.

(a)(4 marks) Will there be any problems with the code above? Why?

(b)(4 marks) If there is a problem, how could you re-write vector_add() to fix it?

Question 7 (8 marks): Translation

Assume a 32-bit virtual address space and further assume that we are using paging. Also assume that the virtual address is chopped into a 20-bit virtual page number (VPN) and a 12-bit offset. The TLB has the following contents in each entry: a 20-bit VPN, a 20-bit PPN, and an 8-bit PID field. This TLB only has four entries, and they look like this:

VPN	PPN	PID
00000	01EFF	00
00000	33A00	01
00010	4AB0A	00
010FF	0A0BC	01

Note all these numbers are in hex. Thus, each digit represents four bits (e.g., hex “F” is “1111”, hex “A” is “1010”, hex “7” is “0111”, and so forth). That is why the 20-bit VPN and PPN are represented by five hex numbers each.

Now, for each of the following virtual address, say whether we have a TLB hit or a TLB miss. IMPORTANT: If it is a hit, provide the resulting physical address (in hex). Note: virtual addresses are also in hex.

(a)(1.5 marks) PID 00 generates the virtual address: 0000000C

(b)(1.5 marks) PID 01 generates the virtual address: 0000000C

(c)(1.5 marks) PID 00 generates the virtual address: 010FFFAA

(d)(1.5 marks) PID 01 generates the virtual address: 010FFFAB

(e)(2 marks) PID 02 generates the virtual address: 010FFFFF

Question 8 (12 marks): TLB and Page faults

TLB misses can be nasty. The following code can cause a lot of TLB misses, depending on the values of STRIDE and MAX. Assume that your system has a 32-entry TLB with a 4KB page size. Assume `int' type has 4 bytes. Assume LRU is used whenever a replacement policy is needed. Assume RAM size is 4MB, and assume the RAM can be entirely used to store the array 'data'.

```
int value = 0;
int data[MAX]; // a big array
for (int j = 0; j < 1000; j++) {
  for (int i = 0; i < MAX; i += STRIDE) {
    value = value + data[i];
  }
}
```

(a)(4 marks) What should you set MAX and STRIDE to so that you can achieve a TLB miss (but *not* a page fault) upon every access to the array 'data'? Explain your answer.

(b)(4 marks) Assume the value of STRIDE is fixed at 4096 (2^{12}), and we change the value of MAX. Please discuss this program's TLB & Memory faulting behavior on different MAX values ranging from 1 to +∞.

(c)(4 marks) Assume the value of MAX is fixed at 64K (2^{16}), and we change the value of STRIDE. Please discuss this program's TLB & Memory faulting behavior on different STRIDE values ranging from 1 to +∞.

Question 9 (14 marks): OS161

Consider the following function in OS161 (from dumbvm.c):

```
1 void as_activate(struct addrspace *as) {
2       int i, spl;
3       spl = splhigh();
4       for (i=0; i<NUM_TLB; i++) {
5         TLB_Write(TLBHI_INVALID(i), TLBLO_INVALID(), i);
6       }
7       splx(spl);
8 }
```

(a)(4 marks) When is this called?

(b)(5 marks) What does the "for" loop from line 4 - 6 do? Why is it needed?

(c)(5 marks) Why are line 3 and 7 necessary?

Question 10 (8 marks): Synchronization II

Your co-worker implements the following code for condition variables (and specifically, the cond_wait() and cond_signal() routines) using **semaphores**:

```
typedef struct __cond_t {
    sem_t s;
} cond_t;

void cond_init(cond_t *c) {
    sem_init(&c->s, 0);
}

// cond_wait(): assumes that the lock ’m’ is held
void cond_wait(cond_t *c, lock *l) {
    unlock(&l); // release lock and go to sleep
    P(&c->s);
    lock(&l); // grab lock before returning
}

void cond_signal(cond_t *c) {
   V(&c->s); // wake up one sleeping waiter (if there is one)
}
```

Does it correctly implement condition variable semantics? Why or why not?

This page is blank. You can use it as an overflow page for your answers.