

**Second Midterm Exam**

October 29, 2018

CS162 Operating Systems

<b>Your name</b>	
<b>SID</b>	
<b>CS162 login (e.g., s162)</b>	
<b>TA Name</b>	
<b>Discussion section time</b>	

**This is a closed book and two 2-sided handwritten notes** examination. You have 80 minutes to answer as many questions as possible. The number in parentheses at the beginning of each question indicates the number of points for that question. You should read **all** of the questions before starting the exam, as some of the questions are substantially more time consuming.

Write all of your answers directly on this paper. ***Make your answers as concise as possible.*** If there is something in a question that you believe is open to interpretation, then please ask us about it!

**Good Luck!!**

<b>Question</b>	<b>Points assigned</b>	<b>Points obtained</b>
<b>P1</b>	<b>12</b>	
<b>P2</b>	<b>16</b>	
<b>P3</b>	<b>18</b>	
<b>P4</b>	<b>19</b>	
<b>P5</b>	<b>17</b>	
<b>P6</b>	<b>18</b>	
<b>Total</b>	<b>100</b>	

## True/False and Why? (12 points)

**MARK THE CHECKBOX NEXT TO YOUR ANSWER.** For each question: 1 point for true/false correct, 1 point for explanation. An explanation **cannot** exceed 2 sentences.

- a) When comparing physical and virtual addresses, the number of offset bits in both addresses must always be the same.

**TRUE**

**FALSE**

Why?

- b) When repeatedly looping through an amount of data larger than the cache size, LRU will yield less page faults than FIFO.

**TRUE**

**FALSE**

Why?

- c) Having a page table fit in a single memory page (frame) reduces external fragmentation.

**TRUE**

**FALSE**

Why?

- d) A multi-level page table is a better solution for a sparse address space than a single level page table approach.

**TRUE**

**FALSE**

Why?

- e) Deadlock will not occur if resources are allowed to be shared up to three times. That is, three threads can acquire the resource before the fourth one tries and blocks.

**TRUE**

**FALSE**

Why?

- f) Banker's algorithm guarantees that all threads eventually receive their requested resources.

**TRUE**

**FALSE**

Why?



### P3. The Kitchen Sync (18 points)

We are given the following implementations for a lock in Pintos, with two threads, A and B, that each run `acquire()` then `release()`. For each implementation, answer the following questions:

- Does the implementation enforce mutual exclusion (Can we ensure one thread starts `release` before the other thread finishes `acquire`)?
- If not, give an interleaving that breaks mutual exclusion. You only need to fill in as many line executions as it takes to break mutual exclusion.
- Is it possible for either A or B to be permanently blocked?
- If yes, which threads can be blocked and on which lines they can be stuck on?

Please make the following assumptions:

- `wait_list` has been initialized already
- `move_thread` is an atomic operation that moves a thread from its current list to the specified list
- Line numbers are specified to the left of each line
- Thread A finishes line 2 before any context switching occurs
- Being “stuck” on a line means that line has started or finished executing, but the next line has not started executing yet
- `thread_block ()` will still work if called with interrupts enabled
- The scheduler skips threads with status `THREAD_BLOCKED`

(Question continues on next page)

(a) (6 points)

```
bool held = false;
struct list wait_list;
```

```
1. acquire() {
2.   if (held) {
3.     move_thread (thread_current (),
4.                 wait_list);
5.   thread_block ();
6. } else {
7.   held = true;
8. }
```

```
9. release() {
10.  if (!list_empty (wait_list)) {
11.    thread_unblock (list_front
12.                  (wait_list));
13.    move_thread (list_front
14.                (wait_list), ready_list);
15.  } else {
16.    held = false;
17.  }
```

Does this enforce mutual exclusion? If not, please indicate an order of line execution that proves this. If needed, we have filled out the first line of execution for you.

Yes  No

Step	1	2	3	4	5	6	7
Running Thread	A						
Line	2						

Is it possible for a thread to be blocked forever? If so, which thread(s) and on what lines(s)?

Yes  No

Thread Blocked				
Line Blocked On				

(b) (6 points)

```
int guard = 0;
bool held = false;
struct list wait_list;
```

```
1. acquire() {
2.   while (test&set (&guard));
3.   if (held) {
4.     move_thread (thread_current (),
5.                 wait_list);
6.     guard = 0;
7.     thread_block ();
8.   } else {
9.     held = true;
10.    guard = 0;
11. }
```

```
12.release() {
13.  while (test&set (&guard));
14.  if (!list_empty (wait_list)) {
15.    thread_unblock (list_front
16.                  (wait_list));
17.    move_thread (list_front
18.                (wait_list), ready_list);
19.  } else {
20.    held = false;
21.  }
22.  guard = 0;
23. }
```

Does this enforce mutual exclusion? If not, please indicate an order of line execution that proves this. If needed, we have filled out the first line of execution for you.

Yes  No

Step	1	2	3	4	5	6	7
Running Thread	A						
Line	2						

Is it possible for a thread to be blocked forever? If so, which thread(s) and on what lines(s)?

Yes  No

Thread Blocked				
Line Blocked On				

(c) (6 points)

```

bool held = false;
struct list wait_list;

1. acquire() {
2.   intr_disable ();
3.   if (held) {
4.     move_thread (thread_current (),
                    wait_list);
5.     thread_block ();
6.   } else {
7.     held = true;
8.   }
9.   intr_enable ();
10.}

11.release() {
12.  intr_disable ();
13.  if (!list_empty (wait_list)) {
14.    thread_unblock (list_front
                      (wait_list));
15.    move_thread (list_front
                  (wait_list), ready_list);
16.  } else {
17.    held = false;
18.  }
19.  intr_enable ();
20.}

```

Does this enforce mutual exclusion? If not, please indicate an order of line execution that proves this. If needed, we have filled out the first line of execution for you.

Yes  No

Step	1	2	3	4	5	6	7
Running Thread	A						
Line	2						

Is it possible for a thread to be blocked forever? If so, which thread(s) and on what lines(s)?

Yes  No

Thread Blocked				
Line Blocked On				

### P4. Deadlock, Deadlock, Deadlock (19 points)

Suppose we have the following total resources and threads T1, T2, T3, and T4 with current allocations and maximum required allocations.

Total Resources		
A	B	C
5	12	7

	Current Allocation			Maximum Allocation		
	A	B	C	A	B	C
T1	1	5	1	3	10	3
T2	0	2	1	2	4	2
T3	2	1	0	2	3	3
T4	1	2	2	2	8	5

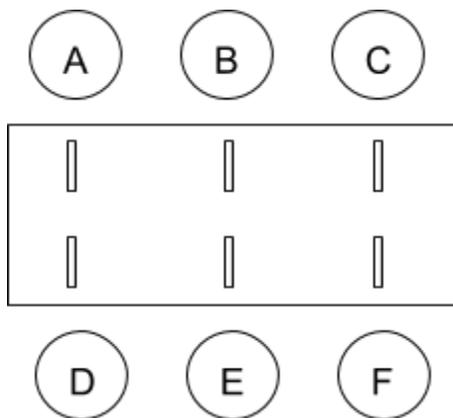
- (a) (5 points) Is the system in a safe state? If so, provide a sequence of resource allocations that would allow all threads to terminate. If not, explain why in two sentences or less.

(b) (4 points) Oh no! We allow threads to acquire resources as they request them and as a result we have gotten ourselves into the deadlocked state, as shown below. We can force a SINGLE thread to release all of its current resources. Choose a SINGLE thread to forcefully release all of its resources in order to ensure that all programs finish or if this is not possible state "No thread". Write the answer in the box. In both cases, provide a short explanation outside of the box (no more than two sentences).

	Current			Maximum		
	A	B	C	A	B	C
T1	2	7	2	3	10	3
T2	0	3	1	2	4	2
T3	1	0	1	2	3	3
T4	1	2	3	2	8	5

Thread:

Now let's consider another problem also dealing with deadlock.



Suppose philosophers A, B, C, D, E, and F are sitting at a rectangular table.

- Philosophers need "n" chopsticks to eat and once done eating will place its chopsticks back to where they belong. Then they try to eat again.
- Each philosopher has one chopstick in front of them.
- Philosophers can use chopsticks that are in front of themselves, their neighbors, or the philosopher across from them. For example in the provided seating, philosopher B can grab A, B, C, or E's chopstick if it is free.
- Philosophers have a preferred set of chopsticks they use. They will try to grab their first preferred chopstick and wait if it is taken and so on.

(c) (5 points) Consider the problem where philosophers need THREE chopsticks to eat ( $n=3$ ). Provide a preference of three chopsticks for each philosopher in order to prevent deadlock or state that it is not possible in the table below. If the "not possible" box is checked, we will NOT grade the answers in the table.

Not possible

		Philosopher					
		A	B	C	D	E	F
Chopstick	1st	A					
	2nd	B					
	3rd	D					

(d) (5 points) Suppose philosophers can now reach diagonally across the table in both directions to grab a free chopstick. For example, philosopher B can try to grab D and F in addition to the previous chopsticks it could grab. Suppose that philosophers now need FOUR chopsticks to eat ( $n=4$ ). Provide a preference of FOUR chopsticks for each philosopher in order to prevent deadlock or state that it is not possible in the table below.

Not possible

		Philosopher					
		A	B	C	D	E	F
Chopstick	1st	A					
	2nd	B					
	3rd	E					
	4th	D					

## P5. Just 'Bout That Translation, Boss (17 points)

Suppose we have a 42 bit virtual address space, a page size of 1KB and a single level page table with a page table entry size of 4 bytes

(a) (3 points) How many bits is the virtual page number? The offset? Place your answer in the box provided

VPN:

Offset:

(b) (3 points) Given that we need at least 9 control bits per PTE, what is the maximum size of our physical address space? Place your answer in the box provided.

c) (8 points) Assume the following PTE format (with 9 control bits) stored in big-endian form in the following page table.

PTE:

PPN	Other (6 bits)	Read/Write	Dirty	Valid
-----	----------------	------------	-------	-------

Page Table:

Address	+0	+1	+2	+3	+4	+5	+6	+7
0x8000	6C	65	67	69	6F	6E	20	6F
0x8008	66	20	62	6F	6F	6D	43	08
0x8010	25	29	31	41	54	72	50	56
0x8018	67	6F	20	68	61	77	6B	73

Using the information above, translate each of the following virtual addresses to physical addresses. Assume that the page table pointer is at 0x8000. Place your final answer in the box provided. If you encounter an error, write **ERROR** in the box.

- 0x00000000B03

- 0x00000001F59

- 0x0000000051B

- 0x000000014B2

d) (3 points) Now suppose we want to transform our single level page table into a multi-level page table. Assuming that every page table is required to fit into a single page, how many total levels of page tables do we need to address the entire virtual address space? Place your answer in the box provided.

## P6. ML Homework (18 Points)

As part of a machine learning homework, Natalie wrote a dense matrix-vector multiplication kernel ( $y=Ax$ ). Natalie's computer has a single-core processor with 1 level of L1 cache. The cache is 4-way associative, with 8-byte cache blocks, an LRU eviction policy, and an overall cache size is 512 bytes. Write operations allocate the relevant block into the cache.

Assumptions:

- The variable  $i, j$  are stored in registers and do not consume memory/cache space.
- We define "hit rate" as the percentage of memory accesses (both read accesses and write accesses) that have a cache hit.
- $A$  is saved as a 2d array starting at the address  $0x10$
- $X$  is saved as an array starting at address  $0x500000010$
- $Y$  is saved as an array starting at address  $0x1000000010$
- $Z$  is saved as an array starting at address  $0x1500000010$
- The matrix  $A$  is stored as a row-major matrix (i.e., rows are stored contiguously in memory)
- Assume each value in the vector or matrix is represented as a 1-byte "integer" (i.e. has integer values between 0 and 255).

You may leave answers in the form of a fraction

Natalie's implementation of her dense matrix-vector multiplication kernel is:

```
for (int j=0; j < N; j++) {
    for (int i=0; i < M; i++) {
        y[i] += A[i][j] * x[i];
        z[i] = i;
    }
}
```

Natalie started by testing her code on a small matrix.

Assume

$N = 8$

$M = 8$

(a) (5 points) Compute the hit-rate of the L1 cache at the end of the program. (you can leave you answer in the form of a fraction). Place your answer in the box provided.

Natalie was happy with her results, so she ran her code on a bigger matrix she got from her friends in the machine learning lab.

The dimensions of the new matrix are:

$N = 128$

$M = 128$

(b) (8 points) For each of the arrays in the problem, state whether its cache hit rate will increase, remain the same, or decrease compared to the rate of the previous part. What miss types (compulsory, capacity, conflict) do each of the array experience?

Array	Increase/Decrease/Same	Miss Types
x		
y		
z		
A		

(c) (5 points) In one sentence, how would you optimize Natalie's code to run faster by obtaining a better cache hit-rate for large matrices?

## P7. Potpurri (0 points)

a. Illustrate your favorite dogspotting post below

b. How are we doing so far?

When you have 162 midterms, projects, and homeworks all within a week of each other

