CSC-501: Operating Systems Principles (Fall 2018)

Please Read All Questions Carefully!
There are 12 total numbered pages.
You have at least 75 minutes.

Please put your FULL NAME and Unity ID on THIS page only.
Please put your Student ID on the right corner of each page with odd page numbers.

Name: _____
Unity ID: _____

Student ID: _____

Grading Page

	Points	Total Possible
Q1		15
Q2		6
Q3		6
Q4		3
Q5		4
Q6		15
Q7		12
Q8		5
Q9		5
Q10		3
Q11		1
Total		75

Student ID: _____

1. Circle True or False for each of the following statements.

(**True**) (False) A system call is a procedure call.

(True) (**False**) A cooperative scheduling approach should not be used in any system,

(**True**) (False) MLFQ learn from history, but it also forgets purposely.

(**True**) (False) Every address a program sees is virtual, and every virtual address needs to be translated by hardware.

(True) (**False**) A virtual memory system that uses paging is vulnerable to external fragmentation.

(True) (**False**) A segmentation fault is caused by some memory access that accesses the stack segment.

(**True**) (False) TLB, just like other caches, takes advantage of temporal locality and spatial locality.

(True) (**False**) If a VPN to PFN mapping is cached in a TLB entry, the TLB valid bit has the same value as the page table valid bit.

(True) (**False**) The return-from-trap for system calls and TLB-miss handlers both jump to the PC after trap.

(True) (**False**) Accesses to different virtual addresses always take the same amount of time.

(True) (**False**) A page fault is caused by some programming mistakes, i.e., bugs.

(True) (**False**) One can use hardware to handle page faults to significant speed up the handling of page faults.

(**True**) (False) The clock algorithm approximates LRU.

(True) (**False**) COW is used by `fork()` to simplify the implementation.

(**True**) (False) In Linux, the kernel portion of a process's virtual address space is the same across processes.

2. Name three key hardware mechanisms that enable limited direct execution, and briefly explain why each of the three is needed.

   a. separation of kernel mode and user mode
   b. privileges instructions, especially trap and return-from-trap
   c. timer interrupt

3. What are the two main performance metrics for scheduling policies?

   Turnaround time and response time

   What is the major design goal of MLFQ as a scheduling policy?

   Balance turnaround time and response time

   What is the major design goal of lottery scheduling as a scheduling policy?

   Fairness or proportional

4. Suppose you do your homework assignments in SJF-order. After all, you feel like you are making a lot of progress! What might go wrong?

   Time-consuming assignments may miss deadlines.

5. Assume the following C code is compiled to create the executable program, `pointless`, which is then run on a UNIX system.

```
int main(int argc, char *argv[]) {
   forkthem(5);
}

void forkthem(int n) {
   if (n > 0) {
      fork();
      forkthem(n - 1);
   }
}
```

How many processes will be part of the `pointless` program? Assuming `fork()` does not return an error.

32

6. This is the multi-level page table question, and it is a smaller scale version of the problem we discussed in class.

Some basic assumptions:

- The page size is an unrealistically-small 16 bytes
- The virtual address space for the process in question (assume there is only one) is 256 pages, or 4 KB
- Physical memory consists of 64 pages

Thus, a virtual address needs 12 bits (4 for the offset, 8 for the VPN). A physical address requires 10 bits (4 for the offset, 6 for the PFN).

The system assumes a multi-level page table. Thus, the upper four bits of a virtual address are used to index into a page directory; the page directory entry (PDE), if valid, points to a page of the page table. Each page table page holds 16 page-table entries (PTEs). Each PTE, if valid, holds the desired translation (physical frame number, or PFN) of the virtual page in question.

The format of a PTE is:
  unused | VALID | PFN5 ... PFN0

It is thus 1 byte with the top bit unused.

The format of a PDE is essentially identical:
  unused | VALID | PT5 ... PT0

For this question, assume the page directory is in page 21. The physical memory dump is on this and the next pages, where your answers will go.

(a) CIRCLE bytes that are accessed during a load from virtual address 0x456.

46 (page 21), 00 (the sixth from page 6)

(b) Put SQUARES around bytes that accessed during a load from virtual address 0x123.

4c (page 21), 45 (page 12), 17 (page 5)

```
page   0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page   1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page   2: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f d8 7f 90
page   3: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page   4: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page   5: 0d 07 0b 17 08 06 1e 16 07 12 14 0f 0b 16 19 12
page   6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page   7: 08 0b 0d 0d 15 16 10 19 18 0f 0e 18 18 19 03 06
Page   8: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page   9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  10: 7f 7f 7f be 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  11: 18 16 09 12 13 11 05 1c 1c 06 06 03 09 1e 09 00
page  12: 1c 4f 45 08 10 19 1c 0e 15 18 13 08 00 17 03 15
page  13: 0e 41 44 03 0d 10 1e 07 14 12 03 16 08 1c 05 1a
page  14: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  15: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  16: 03 10 18 0b 0b 06 14 0a 16 1d 16 0f 16 01 08 07
page  17: 09 12 18 0e 06 0d 1b 02 0a 0c 07 01 0e 13 06 0b
page  18: 05 0a 13 15 15 07 05 0b 1d 00 19 0b 00 0a 16 16
page  19: 11 1e 02 00 0f 0f 1e 03 02 19 11 15 17 05 1d 00
page  20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  21: 4a 4c 01 49 46 00 0c 11 03 13 13 08 0b 09 03 07
page  22: 05 01 08 0f 19 08 1d 16 1e 05 1c 0b 05 1d 16 1b
page  23: 00 18 06 0b 12 02 02 0d 03 0a 12 0d 09 14 1e 19
page  24: 0c 1e 0a 15 0f 0c 13 06 08 07 08 06 1d 14 0e 14
page  25: 1a 0b 0c 08 0e 0a 16 17 1a 0f 18 13 11 1e 1a 16
page  26: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  27: 09 09 1b 19 1d 01 07 0d 10 05 05 1c 0f 05 09 09
page  28: 7f 7f 7f 7f 7f 7f d7 7f 7f 7f 96 7f 7f 7f 7f 7f
page  29: 14 12 08 1b 14 1d 04 05 15 06 00 09 15 14 1d 0b
page  30: 0a 0b 16 0a 1d 0e 0b 00 17 06 0e 14 0e 1d 0c 0b
page  31: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  32: 0c 07 1b 07 14 06 0c 07 07 17 10 03 15 13 11 05
page  33: 1c 1d 16 00 12 1a 0a 09 1b 1e 06 10 02 13 00 09
page  34: 7f 7f 7f 7f 7f 98 7f 7f 7f 7f 7f 7f 8d 7f 7f 7f
page  35: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  36: 7f 7f 7f 7f 7f 7f 7f 7f a1 7f 7f 7f 7f 7f 7f 7f
page  37: 1a 1a 02 1a 07 1d 15 0b 0f 13 0f 11 19 02 0b 1d
page  38: 1c 12 0d 18 02 0e 1a 13 13 03 19 06 1d 09 10 10
page  39: 17 07 06 09 19 14 09 08 11 0f 08 09 07 09 02 18
page  40: 10 09 13 04 0e 06 0e 0e 14 09 09 11 11 0d 0b 01
page  41: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  42: 7f 97 7f 7f 7f 7f 7f 7f b1 7f 7f 7f 7f 7f 7f 7f
page  43: 05 0e 1b 12 1e 05 00 17 08 12 00 09 04 09 12 19
page  44: 19 10 0b 05 0c 15 12 1a 03 08 01 1c 00 12 0c 00
```

```
page  45: cd 7f 7f 7f 7f 7f 7f ff 7f 7f 7f 7f 7f bb 7f
page  46: 13 0f 0e 17 17 07 09 1e 14 1c 10 09 1c 16 13 17
page  47: 13 00 1e 04 13 09 19 1a 0c 12 00 12 03 13 11 1a
page  48: 0a 0e 0a 18 00 12 00 19 11 10 0b 0e 1e 1e 1d 1c
page  49: 1b 0a 09 16 0f 10 14 10 1e 17 10 04 05 1c 0f 09
page  50: 1b 05 01 0f 11 0c 04 19 0c 1e 09 0f 1e 0a 0b 02
page  51: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  52: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  53: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  54: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f9 a6 7f 7f
page  55: 10 12 04 10 04 0a 0f 02 1e 13 1d 1c 1e 18 09 04
page  56: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  57: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  58: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  59: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  61: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  62: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  63: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
```

Next, we're going to talk about what would happen if some bits get flipped in memory.

(c) You hear about a case where bits in the **page directory** get flipped and cause all translations to be invalid. Is this possible? If so, what bits need to be flipped? If not, why not?

Yes.
Flip all valid bits that are currently in 1 to 0.

(d) You also hear about a case where bits are flipped in **page table entries** such that a load to **any** virtual address always returns the value 0x00. Is this possible? If so, how could it happen? If not, why not?

Yes.
Have the PFN pointing to pages containing all "00".

(e) Finally, you hear about a problem where a bit flip leads to a physical address that is greater than 12 bits long. Is this possible? If so, how could it happen? If not, why not?

No.
It will always be 10, 6 from page table entries and 4 from offset. These numbers are determined by OS and hardware, and memory content cannot change these.

7. In this question, we'll examine a string of memory references to virtual pages (inputs), and use detailed system counters (outputs), the details of which are not relevant for this question, to tell us what happened on such references, i.e., whether a particular memory reference was a TLB hit, TLB miss, or a page fault.

(a) Here is the first trace from some system:

```
Input                       Output
--------------------        ------------
load to virtual page 0      Page fault
load to virtual page 1      Page fault
load to virtual page 2      Page fault
load to virtual page 3      Page fault
load to virtual page 0      TLB miss
load to virtual page 1      TLB miss
load to virtual page 2      TLB miss
load to virtual page 3      TLB miss
```

Assuming a TLB with LRU-based replacement, what can we conclude is the size of the TLB? (i.e., the number of entries it holds?)

At most 3

(b) On a different system, we see this trace:

```
Input                       Output
--------------------        ------------
load to virtual page 0      Page fault
load to virtual page 1      Page fault
load to virtual page 2      Page fault
load to virtual page 3      Page fault
load to virtual page 0      TLB hit
load to virtual page 1      TLB hit
load to virtual page 2      TLB hit
load to virtual page 3      TLB hit
```

Again, assuming a TLB with LRU-based replacement, what can we conclude about the size of the TLB?

At least 4

(c) Finally, on some third system we see the following stream:

```
Input                       Output
--------------------        ------------
load to virtual page 0      Page fault
load to virtual page 1      Page fault
load to virtual page 2      Page fault
load to virtual page 3      Page fault
load to virtual page 0      TLB hit
load to virtual page 1      TLB hit
load to virtual page 2      Page fault
load to virtual page 3      TLB miss
```

Assume the TLB can hold two entries, and the entire memory can hold three pages. What can we conclude about the replacement policy used by the OS?

Page replacement policy can be MRU or random, TLB replacement policy can be random or just keep the first two.

(d) Is it possible for an input steam to incur more TLB misses on a 10-entry TLB than an 8-entry TLB? Why or why not?

Yes. Due to Belady's anomaly.

8. A multi-threaded program looks like the following:

```
volatile int counter = 500;
void *worker(void *arg) {
   counter--;
   return NULL;
}
int main(int argc, char *argv[]) {
   pthread_t p1, p2, p3;
   pthread_create(&p1, NULL, worker, NULL);
   pthread_create(&p2, NULL, worker, NULL);
   pthread_create(&p3, NULL, worker, NULL);
   pthread_join(p1, NULL);
   pthread_join(p2, NULL);
   pthread_join(p3, NULL);
   printf("%d\n", counter);
   return 0;
}
```

Assuming `pthread_create()` and `pthread_join()` do not return an error, which outputs are possible? Please write possible or impossible for each of the following values.

503 __ **impossible** _

502 __ **impossible** _

501 __ **impossible** _

500 __ **impossible** _

499 __ **possible** ___

498 __ **possible** ___

497 __ **possible** ___

496 __ **impossible** _

495 __ **impossible** _

0 __ **impossible** _

9. Of the following items, which are stored in the thread control block, which are stored in the process control block, and which in neither? Please write TCB, PCB, or neither for each of the following.

page table pointer ____**PCB**____

page table __**Neither**____

stack pointer __**TCB**_____

ready list __**Neither**____

program counter __**TCB**_____

10. Assume the following implementation of a lock:

```
void init(lock_t *mutex) {
   mutex->flag = 0; // 0 -> lock is available, 1 -> held
}
void lock(lock_t *mutex) {
   while (mutex->flag == 1)
      ; // L2
   mutex->flag = 1;
}
void unlock(lock_t *mutex) {
   mutex->flag = 0;
}
```

Assume 5 threads are competing for a lock with the implementation above. How many threads can possibly acquire the lock? List all possibilities.

1, 2, 3, 4, 5 (or no 5 if just 4 threads)

11. The question has two parts.
    (a) Is this exam simpler than the sample?

    That's what I was hoping for.

    (b) Will you read the project description for "Demand Paging" before the Wednesday lecture?
    (Hint: the URL is https://people.engr.ncsu.edu/gjin2/Classes/501/Fall2018/assignments/PA2/pa2.html)

    Yes.

This is a scratch paper. Free feel to detach and take it with you.