

Midterm 1: Virtualization

Time: 75 minutes Total Questions/Total points: 70

Instructions

This exam is a closed book, closed notes. You will hand the question set and the scantron sheet back to the instructor after the exam is over.

Please fill in your name and student ID on the answer sheet.

When you begin the exam, you will need to agree with the following statements:

1. 1. I understand that this exam must be worked on independently.
2. 2. I will not communicate or collaborate with any others during the exam in any way;

for example:

- I will not chat, text, or post any questions or information.
 - I will not ask for, receive, or provide help to others.
 - I will notify the instructor if I am aware of others violating this policy.
3. I will not discuss the questions with students of the other section.

Unless stated (or implied) otherwise, you should make the following assumptions:

- The OS manages a single uniprocessor (single core).
- All memory is byte addressable.
- The terminology lg means \log_2 .
- 1 byte = 8 bits.
- 1 hexadecimal digit takes 4 bits.
- Page table entries require 4 bytes unless otherwise stated.
- Data is allocated with optimal alignment, starting at the beginning of a page.
- Leading zeros can be removed from numbers (e.g., 0x06 == 0x6).
- Hex numbers are represented with a preceding "0x"; numbers without 0x are decimal.

Table of powers of 2

Power of 2	Bytes	KB	MB
0	1		
1	2		
2	4		
3	8		
4	16		
5	32		
6	64		
7	128		
8	256		
9	512		
10	1,024	1	
11	2,048	2	
12	4,096	4	
13	8,192	8	
14	16,384	16	
15	32,768	32	
16	65,536	64	
17	131,072	128	
18	262,144	256	
19	524,288	512	
20	1,048,576	1,024	1
21	2,097,152	2,048	2
22	4,194,304	4,096	4
23	8,388,608	8,192	8
24	16,777,216	16,384	16
25	33,554,432	32,768	32
26	67,108,864	65,536	64
27	134,217,728	131,072	128
28	268,435,456	262,144	256
29	536,870,912	524,288	512
30	1,073,741,824	1,048,576	1,024
31	2,147,483,648	2,097,152	2,048
32	4,294,967,296	4,194,304	4,096

Part 1: CPU virtualization short answers

1. Which of these are application benefits of an operating system?
 - a. A set of simpler abstractions against which to program
 - b. Independence from specific hardware and devices
 - c. More control over how hardware is used
 - d. A & B – with an OS, the app has less control over hardware**
 - e. All the above
2. Which of these are **not** user benefits of an operating system?
 - a. Protecting running programs against the misbehavior of other programming
 - b. More physical memory available to programs – OS uses some physical memory**
 - c. Sharing hardware by running multiple programs at the same time
3. Processes are a stream of execution and execution state
 - a. True – this is the definition**
 - b. False
4. Which of the following is not a valid process state
 - a. Blocked
 - b. Ready
 - c. Preempting**
 - d. Waiting
5. Which of the following are present in processes but not programs
 - a. Stack & registers**
 - b. Heap & data
 - c. Code & registers
 - d. Data & code
6. The state of a process is its execution state (registers) and memory state
 - a. True
 - b. False – there is also OS state such as open files**
7. With direct execution, the operating system **can't** take control of the CPU at any time
 - a. True – OS can take control with limited direct execution**
 - b. False

8. What does a processor do if a process executes a privileged instruction in user mode?
- a. **The processor traps**
 - b. The instruction is ignored
 - c. The computer reboots
9. It is safe to let user-mode code enable and disable interrupts
- a. True
 - b. **False – process could prevent OS from regaining control**
10. Which of the following are operations allowed in user-mode execution?
- a. Access to all areas of physical memory – prevented to stop processes from interfering with another process
 - b. **Saving/restoring registers**
 - c. Issuing input/output commands – prevented to ensure processes don't have direct access to I/O hardware
 - d. Modifying processor memory management state – prevented to stop
11. On a timer interrupt, the OS is responsible for transitioning from user to kernel mode.
- a. True
 - b. **False – the hardware is responsible for it**
12. With cooperating multitasking, context switches only happen when a program calls yield()
- a. True
 - b. **False – they can also happen anytime the process makes a system call**
13. On a timer interrupt, the operating system always context switches to a new process
- a. True
 - b. **False – the timeslice of a process maybe longer than timer tick, so same process might be chosen again.**
14. On a context switch, user-mode registers are saved in the process-control block (PCB)
- a. True
 - b. **False – they are stored on the kernel stack**

15. Processes **can't** move from the running state directly to the blocked state
- a. True
 - b. **False – they can when they do I/O**
16. Processes can move from the blocked state directly to the ready state
- a. **True – once I/O completes**
 - b. False
17. Multiple processes can be in the RUNNING state
- a. True
 - b. **False – not on a uniprocessor**
18. You are designing a word processor and want to make sure it responds quickly when a key is struck. Which scheduling goal are you most interested in?
- a. Turnaround time
 - b. **Response time – this is how fast process can respond to a keystroke**
 - c. Fairness
 - d. No starvation
19. You are running a mix of a word processor and a scientific calculation. Which policy do you prefer?
- a. Round robin - **this lets the word processor run right away. Also, you don't know how long the CPU burst is so you can't use SJF or SCTF**
 - b. FIFO/FCFS
 - c. SJF
 - d. SCTF
20. SCTF **doesn't** suffer from head-of-line blocking
- a. **True – it allows for preemption if short jobs arrive**
 - b. False
21. SCTF suffers from starvation
- a. **True – with a lot of short jobs, a long job may never complete**
 - b. False
22. With FIFO scheduling, when a task wakes up from I/O it loses its old place in the ready queue
- a. **True – it goes to the end of the line**
 - b. False

23. With MLFQ, after running tasks in one queue, the scheduler starts running tasks in the next lower-priority queue

- a. **True – one of the MLFQ rules**
- b. False

24. With MLFQ, Boost gives a running process an extra-long time slice

- a. True
- b. **False – it gives higher priority for long running processes**

25. With MLFQ, tasks with the same, short burst length will run repeatedly in the same queue using round-robin.

- a. **TRUE – one of the MLFQ rules**
- b. FALSE

26. Which of these is a difference between MLFQ without boost and SCTF?

- a. **SCTF prioritizes jobs nearly done with their CPU burst**
- b. MLFQ is non-preemptive
- c. MLFQ does not suffer from starvation
- d. MLFQ prioritizes shorter jobs

27. Suppose you have a process A running. It then does this sequence of operations

`Calls fork to create process B()`

`Process A issues an I/O request to the disk`

`The system then continues for a little longer`

What states are Process A and Process B in?

- a. A: Ready and B: Running
- b. A: Blocked and B: Ready
- c. **A: Blocked and B: Running – process A is blocked on I/O so B is moved off Ready queue and starts running**
- d. A: Running and B: Blocked

28. Suppose you have a system using lottery scheduling and want to prioritize interactive jobs (short burst) to let them run quickly, but let batch jobs (long CPU burst) use the CPU otherwise. Which of these ticket allocations makes the most sense?

- a. **Short jobs: 10 tickets, Long jobs: 1 ticket – giving more tickets to interactive jobs will give them a higher chance of running**
- b. Short jobs: 10 tickets, Long jobs: 10 tickets
- c. Short jobs: 1 ticket, Long jobs: 1 ticket
- d. Short jobs: 1 ticket, Long jobs: 10 tickets

29. Given a system using lottery scheduling with 2 jobs. Job A has 8 tickets, job B has 2 tickets. Job B runs continuously, while job A runs half the time. After running the system for a long time, what fraction of the overall runtime does each job have?
- A: 80%, B: 20%
 - A: 50%, B: 50%
 - A: 40%, B: 60% -- job A will run 80% of the time it is active (which is 50% of overall time). Hence job A runs for 40%. Job B runs the remaining time.**
 - A: 20%, B: 80%

Part 2: CPU Virtualization calculations

30. Assume a parent process always continues running before a child process runs, and children run in order created. Parent has PID 1, child PIDs start at 2 and are allocated sequentially. For this code:

```
int child1_pid = fork();
int child2_pid = fork();
printf("c1 = %d, c2 = %d\n", child1_pid, child2_pid );
```

What is printed **first**?

- c1 = 2, c2 = 3 Parent runs first so the print statement prints child PIDs**
 - c1 = 0, c2 = 4
 - c1 = 2, c2 = 0
 - c1 = 0, c2 = 0
31. For the code in the previous question, what is printed **second**?
- c1 = 2, c2 = 3
 - c1 = 0, c2 = 4 – When first child runs, c1 is zero in the child process. c1 forks another child process and prints 4 (its child PID)**
 - c1 = 2, c2 = 0
 - c1 = 0, c2 = 0

32. For this schedule using **FIFO**, what is the average **turnaround** time?

- Less than 3 ms
- Between 3-4 ms
- Between 5-6 ms**
- Between 11-12 ms

Job	Length	Arrival
1	5 ms	12 ms
2	4 ms	10 ms
3	2ms	11 ms

33. For this scheduling using **SJF**, what is the average **response** time?

- a. **Between 2-3ms**
- b. Between 5-6ms
- c. Between 7-8ms
- d. More than 8 ms

Job	Length	Arrival
1	2 ms	0 ms
2	7 ms	0 ms
3	4 ms	0 ms

34. For this scheduling using SCTF, when does job 3 finish?

- a. At 4 ms
- b. At 9 ms
- c. **At 13 ms**
- d. At 21 ms

Job	Length	Arrival
1	2 ms	3 ms
2	7 ms	0 ms
3	4 ms	5 ms

35. Using Round Robin scheduling with a 2ms time slice (quantum), when does job 2 finish?

- a. At 7ms
- b. At 8ms
- c. **At 12ms**
- d. At 14ms

Job	Length	Arrival
1	2 ms	1 ms
2	7 ms	0 ms
3	3 ms	5 ms

Part 3: Memory Virtualization short answers

36. Which mechanism provides the least protection?
- a. **Static relocation – process can access anything in memory**
 - b. Base and bounds
 - c. Segments
 - d. Paging
37. Which mechanism has the least wasted memory for a standard process layout?
- a. **Segments – segment tables takes very little space (compared to page tables)**
 - b. Base and bounds – no, because it wastes space between stack and the heap
 - c. Paging with linear page tables – linear page tables can be large
38. Which mechanisms support sharing?
- a. Segments
 - b. Paging
 - c. Paging and base/bounds
 - d. Base/bounds and segments
 - e. **Segments and paging - processes can both refer to the same segment, or have a page table that refer to the same page**
39. What is the major benefit of paging over segmentation?
- a. Less wasted memory – page tables can waste memory
 - b. Faster performance – can be slower due to pagetable lookup
 - c. **Less external fragmentation – all pages are the same size**
 - d. Less MMU state – not much smaller
40. For a single sequential pass over data, a 4-entry TLB performs better than a 1-entry TLB
- a. True
 - b. **False -- with a single sequential, the process only accesses 1 TLB entry at a time so it doesn't need more than 1.**
41. To translate a virtual address using paging, the MMU adds the offset to the physical page number from the TLB
- a. True

b. False – it appends the offset and doesn't add the offset

42. With multi-level page tables, larger pages lead to fewer levels in the page table

a. True – larger pages imply more bits used for offset

b. False

43. A process can't access the data of another process because the other process's addresses are marked invalid in the page table.

a. True

b. False – each process has a different page table

44. With segmentation, the address space of each process **doesn't** need to be allocated contiguously in physical memory

a. True

b. False – each segment is continuous but the whole address space is not

45. When using paging, larger pages causes more external fragmentation

a. True

b. False – large pages cause more internal fragmentation

46. Compared to using segments, using a 2-level page table with no TLB doubles the number of memory references

a. True

b. False – it triples it (from 1 reference to 3 references)

47. When the dirty bit is set in a PTE, it means the contents of a TLB entry do **not** match the page table

a. True

b. False – it means contents in memory don't match contents on disk

48. FIFO page replacement always has more page faults than LRU page replacement

a. True

b. False – depends on the workload pattern

49. To use huge pages (such as 2mb pages), a programmer always has to request them when allocating memory

a. True

b. False – OS can transparently allocate huge pages.

50. With shared memory, processes map the same physical pages to the same virtual addresses in multiple processes
- a. True
 - b. **False – they can be mapped to different virtual addresses**
51. An LRU TLB replacement policy is worse than a random TLB replacement policy for sequential workloads.
- a. **True – LRU does badly for sequential workloads as it tends to always evict things before they are needed, while random may leave some in the TLB**
 - b. False
52. With ASIDs, there is no performance penalty for context switching when using a TLB
- a. True
 - b. **False - there are still extra TLB misses from running different code**
53. With a software-filled TLB, the OS specifies the layout of the page table
- a. **True – the hardware just traps into the OS and the OS can use whatever datastructure it wants**
 - b. False
54. When the present bit is cleared in a PTE, accessing the associated page will cause a segmentation fault
- a. True
 - b. **False – it will lead to a page fault**
55. Demand paging never loads the data at an address referenced by a process before it is used the first time
- a. True
 - b. **False - it loads other addresses on the same page as the faulting address during a page fault**
56. If the clock hand in clock page replacement is moving quickly, it could mean
- a. The program has high locality
 - b. **The program has low locality - lots of pages have been referenced and skipped over**
57. With clock page replacement, a page that is marked dirty always has the referenced bit set
- a. True

- b. **False - you can have modified the page at one time and then not-referenced it since the clock hand last passed by**

58. When a system is thrashing, the CPU is fully utilized

- a. True
- b. **False - the CPU is not very busy because it is waiting to write or read pages from storage to service page faults .**

59. With inverted page tables, every physical page in a system has a PTE

- a. **True – this is the definition**
- b. False

60. Copy-on-write is faster than regular copy when a process reads from most of the copied pages.

- a. **True - we don't have to copy anything. If a process wrote to most of the pages it would be more expensive**
- b. False

Part 3: Memory Virtualization calculations

61. Given a virtual address size of 32 bits and a page size of 256 bytes and a PTE size of 4 bytes, what is the size of a linear page table?

- a. 4MB
- b. 16MB
- c. **64MB - VPN has 2^{24} bits * 2 bytes/pte = 2^{26} bytes = 64mb**
- d. 256MB

62. Given a virtual address size of 32 bits and a page size of 256 bytes and a PTE size of 4 bytes, how many levels of page table are needed?

- a. 1
- b. 2
- c. 3
- d. **4 - 24 bits to translate, 6 bits per level = 4 levels**

63. Given a virtual address size of 32 bits and a page size of 256 bytes and a PTE size of 4 bytes, what is the **smallest number of pages needed for page tables** for a process with a standard layout and 1kb of code, 1 kb of data, 1kb of heap, and 1kb of stack

- a. 1
- b. 3
- c. 5
- d. **7 - You have 4 levels and need to map 8 pages at the start of the address space and 4 at the end, so you need 1 inner page table at each end (2), 1 2nd level table at each end (2), 1 3rd level pages table at each end (2), and 1 outermost table.**

64. Given a 4 entry TLB and 4KB pages, what will the TLB miss rate be for the line of code shown in boldface?

- a. **Between 0 and 1%**
There will be 1 TLB miss every 512 references
- b. Between 1 and 5 %
- c. Between 5 and 10%
- d. Between 10 and 25%
- e. Higher than 25%

```
// doubles are 8 bytes
double array[1024*1024];
double sum = 0;
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
        Sum = sum + array[1024*i+j];
```

65. Consider this variation of the code. Given a 4 entry TLB and 4KB pages, what will the TLB miss rate be for the line of code shown in boldface?

- a. Between 0 and 1%
- b. Between 1 and 5 %

```
// doubles are 8 bytes
double array[1024*1024];
double sum = 0;
for (i = 0; i < 1024; i++)
    for (j = 0; j < 1024; j++)
        Sum = sum + array[i+1024*j];
```

- c. Between 5 and 10%
- d. Between 10 and 25%
- e. **Higher than 25% - Each reference goes to a different page, so all references will miss**

66. Using the same variation of the code from the previous question, consider a system with a 16 entry TLB and 2MB pages. What is the TLB miss rate for the line of code in boldface when running this loop many times?

- a. **Between 0 and 1 % - the data structure is 8MB, so it takes 4 TLB entries and they all fit in TLB**
- b. Between 1 and 5 %
- c. Between 5 and 10%
- d. Between 10 and 25%
- e. Higher than 25%

67. With 3 pages of memory, demand paging, FIFO replacement, and this reference string, what pages will be in memory at the end? 2, 1, 3, 5, 2, 4, 1, 2, 3, 4

- a. **1, 3, 4 - the fault string is: 2, 1, 3, 5 (E:2), 2 (E:1), 4 (E:3), 1 (E:5), 2, 3 (E: 2), 4**
- b. 2, 3, 4
- c. 2, 1, 3
- d. 4, 2, 1

	Memory
	2, 1, 3
5	5, 1, 3
2	5, 2, 3
4	5, 2, 4
1	1, 2, 4
2	1, 2, 4
3	1, 3, 4
4	1, 3, 4

68. With 3 pages of memory, demand paging LRU replacement and this reference string, assuming memory starts empty, how many page faults will there be? 4, 3, 4, 1, 3, 2, 3, 0

- a. 3
- b. 4
- c. **5**
- d. 6
- e. 7

4 (M)	4
3 (M)	4, 3
4 (H)	4, 3,
1 (M)	4, 3, 1

3 (H)	4, 3, 1
2 (M)	2, 3, 1
3 (H)	2, 3, 1
0 (M)	2, 3, 0

69. In the code shown below, where is `longLine` stored?

- a. Static data
- b. Heap
- c. **Stack – longline is a local variable**
- d. Code

70. In the code shown below, where is `maxlen` stored?

- a. **Static data - global**
- b. Heap
- c. Stack
- d. Code

```
int maxlen = 512;

int main(int argc, char **argv) {
    char shortLine[10];
    char * longLine = malloc(maxlen);
    parseArgs(argc, argv, shortLine, longLine);
}
```