CS 537: Introduction to Operating Systems
Fall 2023: Midterm Exam #1

This exam is closed book, closed notes.

No calculators may be used.

You have 1 hour 15 minutes to complete this exam.

Write all of your answers on the accu-scan form with a #2 pencil:

- CANVAS LAST NAME - fill in your last (family) name starting at the leftmost column
- CANVAS FIRST NAME - fill in the first five letters of your first (given) name
- IDENTIFICATION NUMBER - This is your UW Student WisCard ID number
- ABC of SPECIAL CODES - Write your lecture number as a three digit value 001, 002, or 010.

These exam questions must be returned at the end of the exam, but we will not grade anything in this booklet.

You may separate the pages of this exam if it helps you.

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 $\log_2$
- 2^{10} bytes = 1KB
- 2^{20} bytes = 1MB
- Page table entries require 4 bytes
- Data is allocated with optimal alignment, starting at the beginning of a page
- Assume leading zeros can be removed from numbers (e.g., 0x06 == 0x6).
- Hex numbers are represented with a proceeding “0x”

This exam has 60 questions. Each question has the same number of points.

Good luck!

Virtualizing the CPU

Designate if the statement is True (A) or False (B).

1. The **CPU dispatcher** determines the policy for which process should be run and when.
False, The scheduler determines which process runs and when.
2. With **cooperative multitasking**, it is possible for a process to keep running on the CPU for as long as it chooses.
True, in cooperative multitasking a process must relinquish the CPU as the scheduler cannot preempt a running process.
3. When executing the **return-from-trap** instruction, hardware restores the user process's registers from the kernel stack, changes to user mode, and jumps to a new code location.
True, the OS issues the return-from-trap instruction and the steps the hardware does these steps when executing the instruction.
4. When **an I/O operation completes**, the previously blocked process moves into the RUNNING state.
False, A previously running process moves from the BLOCKED state to the RUNNABLE (READY) state. The scheduler decides which process moves to the RUNNING state from those that are RUNNABLE.
5. The **fork**() system call clones the calling process and executes a new program in the child process.
False, The fork() system call clones the calling process but it does NOT start a new program in the child process. That is the exec() system call.
6. A **FIFO scheduler** has lower average turnaround time when long jobs arrive after short jobs, compared to when short jobs arrive after long jobs.
True, the jobs that come first will be scheduled first (thus having a lower turnaround time for them) and the overall average would then be lower.
7. An **SJF scheduler** may preempt the currently running job.
False, SJF is a non-preemptive scheduler.
8. An **SJF scheduler** can suffer from the convoy effect where a lot of short-running jobs can get stuck behind a long-running job.
True, Jobs that arrive after a long job has already been scheduled would need to wait, since SJF is non-preemptive.
9. A **RR scheduler** may preempt the currently running job.
True, RR is a preemptive scheduler.
10. An **STCF scheduler** cannot cause jobs to starve.
False, if a constant stream of short-running jobs arrive then a long-running job will constantly be put off and never scheduled (i.e. starve).
11. If all jobs arrive at the same time, an SJF and an STCF scheduler will behave the same.
True, since all jobs arrive at the same time, STCF will behave like SJF.
12. A **process** is another name for a program.
False, A program is a passive collection of instructions, a process is a running(executing) version of those instructions.
13. When the OS relinquishes control of the CPU to a process, it retains control by setting the kernel bit.
False, The OS maintains control of the CPU by having a timer-interrupt which returns control of the CPU to the OS and restricting the instructions the process can run.
14. The OS provides the initial arguments to a new process by placing them in the heap.
False, The OS places the arguments in the process's stack.
15. **Limited Direct Execution** consists of privilege levels for user and kernel modes and a timer interrupt and handler to regain control of the CPU.
True, LDE does include these two aspects as well as a handler for the timer-interrupt.
16. A **trap** into a system call switches from user to kernel modes.
True, the trap instruction does switch from user mode to kernel mode and jump to a trap handler.
17. A **context switch** occurs when a process performs a system call.

False, A context switch occurs when one process is preempted and a different process is scheduled.

18. Different processes accessing the same address within their virtual address space see the same contents.

False, even through the virtual addresses may be the same, they refer to different physical addresses.

19. The **exec()** call returns a value of zero on success.

False, The exec() call on success does not return.

20. The purpose of **boosting in MLFQ** is to ensure no processes starve.

True, lower-priority processes would receive no CPU time if higher-priority processes consume all the CPU time.

Select the one best answer, A - E.

```
Consider three processes with the following execution trace.  All IO takes
5 time slices and each process needs 3 CPU time slices to complete running.

./process-run.py -l 3:50,3:50,3:50

Time        PID: 0        PID: 1        PID: 2           CPU          IOs
  1         RUN:io         READY         READY             1
  2        BLOCKED       RUN:cpu         READY             1            1
  3        BLOCKED        RUN:io         READY             1            1
  4        BLOCKED       BLOCKED       RUN:io              1            2
  5        BLOCKED       BLOCKED      BLOCKED                           3
  6        BLOCKED       BLOCKED      BLOCKED                           3
  7*   RUN:io_done       BLOCKED      BLOCKED             1            2
  8         RUN:io       BLOCKED      BLOCKED             1            2
  9*       BLOCKED   RUN:io_done      BLOCKED             1            2
 10*       BLOCKED       RUN:cpu        READY             1            1
```

21. What will the state of process 0 be in the next time slice?

A. DONE B. BLOCKED C. RUN:io_done D. RUN:io E. READY

B, each io takes 5 time slices. Process 0 has only been blocked for 2, so it must still be in the BLOCKED state.

22. What will the state of process 1 be in the next time slice?

A. DONE B. BLOCKED C. RUN:io_done D. RUN:io E. READY

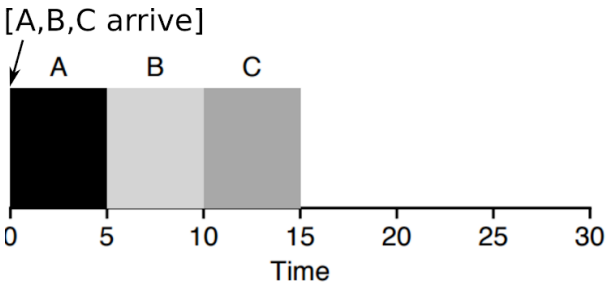
A, Each process needs 3 CPU time slices. This process has had 3 CPU time slices. It is DONE.

23. What will the state of process 2 be in the next time slice?

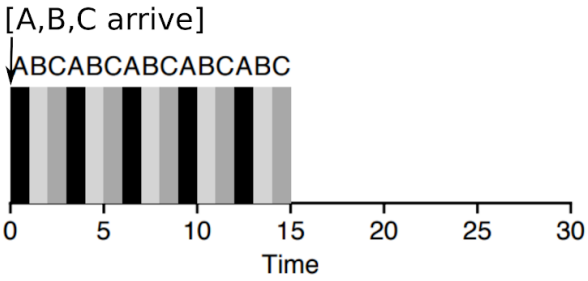
A. DONE B. BLOCKED C. RUN:io_done D. RUN:io E. READY

C, The current state of the process is READY and it is returning from an IO call. Its state will be RUN:io_done.

Consider the following graphs showing the scheduling of three processes, A, B, and C using different scheduling policies.



FCFS



Round robin every 1s

24. What is the turnaround time for process B using FCFS?
A. 5 B. 7.5 C. 10 D. 12.5 E. 15
C, Completion time (10) - Arrival time (0) = 10
25. What is the response time for process B using FCFS?
A. 5 B. 7.5 C. 10 D. 12.5 E. 15
A, First Run Time (5) - Arrival time (0) = 5
26. What is the turnaround time for process C using RR?
A. 5 B. 7.5 C. 10 D. 12.5 E. 15
E, Completion time (15) - Arrival time (0) = 15
27. What is the average response time using FCFS?
A. 1 B. 5 C. 10 D. 14 E. 15
B, (0 + 5 + 10) / 3 = 5
28. What is the average time to completion using RR?
A. 1 B. 5 C. 10 D. 14 E. 15
D, (13 + 14 + 15) / 3 = 14
29. Which of the following is NOT possible for the MLFQ scheduler?
A. MLFQ learns which jobs are long running.
B. MLFQ may starve jobs.
C. MLFQ may use different length time slices for jobs
D. MLFQ may use round robin
E. MLFQ may forget what it has learned about jobs
B, MLFQ moves jobs down queues learning they are long-running, it may use different length time slices for different queues, it uses round robin for jobs in the same queue, and during boosting it forgets what it has learned about jobs. Starvation is impossible due to boosting.
30. Which of the following is NOT true about a MLFQ scheduler?
A. It can preempt running jobs
B. It may change the priority of a process if the boost time has transpired
C. Jobs with higher priority preempt jobs of lower priority
D. It knows the runtime of jobs before scheduling them
E. Jobs with the same priority run using RR
D, MLFQ does not have an oracle to know the runtime of a job. If a job uses up its time slice then it is demoted.

Virtualizing Memory

Designate if the statement is True (A) or False (B).

31. One goal in **memory virtualization** is to ensure that the OS's method of virtualization is transparent to the processes.
True, transparent means the user-level process is unaware of the fact that the OS is virtualizing addresses.
32. The major difference between **static and dynamic relocation** is who converts a virtual address to a physical address -- The OS or hardware.
False, The major difference is when the conversion takes place. In static relocation virtual addresses are converted to physical addresses when the program is loaded into memory. In dynamic relocation addresses are converted as the process runs.
33. One **shortcoming of static relocation** is that it provides no protection of one process accessing memory of another process.
True, there is no hardware support to limit what addresses are accessible.
34. **Base+Bounds** is one type of static relocation.
False, Base+Bounds is a dynamic relocation method.
35. **Segmentation** enables sparse allocation of an address space.

True, the addresses between the heap and the stack do not need to be allocated as the bounds registers for each segment will identify if a virtual address is going outside of the segment's range.

36. **Base+Bounds** provides no protection of one process accessing memory of another process.

False, If a process attempts to access outside its bounds an interrupt occurs.

37. In **segmentation**, to convert a virtual address to a physical address, the offset is always added to the base register for the segment.

False, Only after checking the bounds is the offset combined with the base register and for segments that grow negatively, the negative offset is added.

38. The **address space** is another name for physical memory.

False, It is how physical memory is presented to the running process as a continuous range of addresses beginning from 0x0.

39. In an address space of 16 KB with a page size of 4 KB, the VPN uses 2 bits of the virtual address.

True, 16 KB address space needs 14 bits to represent an address and a 4 KB page means 12 bits for the offset, leaving 2 bits for the VPN.

40. One of **paging's major advantages** is that it makes memory management easier because all of physical memory is divided into fixed sized units called virtual page numbers.

False, The fixed sized units of physical memory are called page frames.

41. Using **ASIDs in the TLB** means fewer TLB misses may occur after a context switch.

True, Without the ASID all entries would be marked invalid causing all new translations to cause a TLB miss.

42. In a **hardware-managed TLB**, the MMU raises an exception on a TLB miss.

False, The hardware would lookup the PTE in memory and record it in the TLB.

43. The **valid bit** in the TLB indicates if there is a mapping between the VPN and the PFN. If it is not set then a segmentation fault should be raised.

False, The valid bit in the TLB indicates if the entry in the TLB is a true mapping between VPN to PFN. If it is not set then the page table in physical memory needs to be consulted.

44. The **present bit** in the PTE indicates if the mapping between the VPN and the PFN is present in the page table. If it is not present then a segmentation fault should be raised.

False, The present bit indicates if the page is currently loaded into memory. If it is not set then a page fault is raised.

45. A **page fault** occurs when a process tries to access a page of memory that is outside its address space.

False, A page fault occurs when a process tries to access a page that is currently swapped out to the disk.

46. When converting a virtual address to a physical address using a **multi-level page table** the first thing the MMU consults is the page directory.

False, The first thing consulted is the TLB.

47. In a **multi-level page table** the VPN is divided into a portion for the page directory index and a portion for indexing into a page of the page table.

True, The top bits are for the page directory index and the middle bits are for the index into the page of the page table.

48. In a paging memory model, if the **page size increases** but the address space stays the same size then the number of entries in the page table must increase.

False, Having a fixed address space means that an address is fixed in size and having a larger page size means more bits are used for the offset within a page. This means there are fewer bits for the VPN, fewer pages, and fewer entries in the page table.

49. The **clock algorithm** attempts to approximate the least-recently used algorithm.

True

50. **Thrashing** is when memory demands exceed physical memory and the system is constantly paging.

True

Select the one best answer, A - E.

Segment Base/Bounds Translation

Consider a virtual address space of size 1K and physical memory of size 16K with the base and limit(bounds) registers for each segment as shown below:

Segment 0 base (grows positive): 0x00001603 (decimal 5635)
Segment 0 limit : 274

Segment 1 base (grows negative) : 0x00002468 (decimal 9320)
Segment 1 limit : 408

Convert each virtual address to its physical address or select segmentation violation if it is out-of-bounds. For these problems, you should assume a simple address space with two segments: the top-bit of the virtual address can thus be used to check whether the virtual address is in segment 0 (topbit=0) or segment 1 (topbit=1).

51. Virtual Address: **0x017f (decimal: 383)**
A. 0x167f (decimal: 5759)
B. 0x1768 (decimal: 5992)
C. 0x17c2 (decimal: 6082)
D. segmentation violation
E. none of the above

D

52. Virtual Address: **0x02f5 (decimal: 757)**
A. 0x235d (decimal: 9053)
B. 0x255d (decimal: 9565)
C. 0x275d (decimal: 10077)
D. segmentation violation
E. none of the above

A

53. Virtual Address: **0x03e7 (decimal: 999)**
A. 0x255d (decimal: 9565)
B. 0x17c2 (decimal: 6082)
C. 0x244f (decimal: 9295)
D. segmentation violation
E. none of the above

C

Size of Linear Page Table

What is the size of a linear page table for one process given the following assumptions? Remember, unless otherwise specified, assume PTEs are 4 bytes each.

54. **Bits in virtual address: 20, Page size: 4KB**
A. 2KB
B. 512 bytes
C. Situation not possible
D. Not enough information to determine page table size
E. None of the above

A

55. **Bits in virtual address: 20, Number of Physical Pages: 1024**
A. 4KB
B. 16KB
C. Situation not possible
D. Not enough information to determine page table size
E. None of the above

D

Multi-level Page Tables

Page Sizes are 32 bytes
Virtual Address space is 1024 pages (or 32 KB)
Physical Memory consists of 128 pages
A Virtual Address needs 15 bits (5 for the offset, 10 for the VPN)
A Physical Address needs 12 bits (5 for the offset, 7 for the PFN)

The system uses a multi-level page table. The upper five bits of a VA are the index into the page directory to get the PDE. If the PDE is valid, it points to a page of the page table. Each page holds 32 PTEs. If the PTE is valid, it holds the desired translation (PFN).

The format of a PTE is:
VALID | PFN6 ... PFN0

The format of a PDE is identical:
VALID | PT6 ... PT0

The PDBR holds the value: 0x51 (decimal: 81) [This means the page directory is held in this page]
The memory dump of the pages of memory are on the last pages of the exam.

56. When accessing the **virtual address 0x65d8**, what will be the first page accessed?
A. 0x65 (decimal: 101)
B. 0x19 (decimal 25)
C. 0xd8 (decimal 216)
D. 0x51 (decimal: 81)
E. Error or None of the above

D

57. What **index of the page directory** will be accessed?

A. 0x19 (decimal: 25)
B. 0x06 (decimal: 25)
C. 0x65 (decimal: 101)
D. 0x05 (decimal: 5)
E. Error or None of the above

A

58. What will be the **second page accessed**?
A. 0x0e (decimal: 14)
B. 0x65 (decimal: 101)
C. 0x7f (decimal: 127)
D. 0x25 (decimal: 37)
E. Error or None of the above

C

59. What are the **contents of the corresponding PTE** that will be read?
A. 0x7f (decimal: 127)
B. 0xfa (decimal: 250)
C. 0x00 (decimal: 0)
D. 0x93 (decimal: 147)
E. Error or None of the above

A

60. What is the **final physical address** for this virtual address?
A. 0xff8 (decimal: 4088)
B. 0xf38 (decimal: 3896)
C. 0x2b8 (decimal: 696)
D. 0x018 (decimal: 24)
E. Error or None of the above

E