CS 537: Introduction to Operating Systems
Fall 2016: Midterm Exam #1

This exam is closed book, closed notes.

All cell phones must be turned off and put away.

No calculators may be used.

You have two hours to complete this exam.

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

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

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

- The OS manages a single uniprocessor
- 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).

This exam has multiple versions. To make sure you are graded with the correct answer key, you must identify this exam version with a Special Code in Column A on your accu-tron sheet. Be sure to fill in the corresponding bubble as well. Your special code is

Good luck!

Part 1: Virtualizing the CPU [3 points each]

Designate if the statement is True (a) or False (b).

1) The **CPU dispatcher** determines the policy for which process should be run when.
2) With **cooperative multitasking**, it is possible for a process to keep running on the CPU for as long as it chooses.
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.
4) When an **I/O operation completes**, the previously blocked process moves into the RUNNING state.
5) The **fork()** system call clones the calling process and overlays a new file image on the **child process**.
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.
7) An **SJF scheduler** may preempt the currently running job.
8) An **SJF scheduler** can suffer from the convoy effect.
9) An **RR scheduler** may preempt the currently running job.
10) An **STCF scheduler** cannot cause jobs to starve.
11) If all jobs arrive at the same point in time, an SJF and an STCF scheduler will behave the same.
12) If all jobs have identical run lengths, a RR scheduler provides worse average turnaround time than FIFO.
13) With an **MLFQ scheduler**, jobs run to completion as long as there is not a higher priority job.
14) With an **MLFQ scheduler**, while a job is waiting for I/O to complete, the job remains in the READY state without changing priority.
15) With **lottery scheduling**, ready jobs must be sorted by the number of tickets that they hold.

Part 2: Process States [1 points each]
Assume you have a system with three processes (A, B, and C) and a single CPU. Assume an MLFQ scheduler. Processes can be in one of five states: RUNNING, READY, BLOCKED, not yet created, or terminated. Given the following cumulative timeline of process behavior, indicate the state the specified process is in AFTER that step, and all preceding steps, have taken place.

For all questions in this Part, use the following options for each answer:

a. RUNNING
b. READY
c. BLOCKED
d. Process has not been created yet
e. Process has been terminated

Step 1: Process A is loaded into memory and begins; it is the only user-level process in the system.

16) Process A is in which state?

Step 2: Process A calls fork() and creates Process B. Process B is scheduled.

17) Process A is in which state?
18) Process B is in which state?

Step 3: The running process issues an I/O request to the disk.

19) Process A is in which state?
20) Process B is in which state?

Step 4: The running process calls fork() and creates process C. Process C is not yet scheduled.

21) Process A is in which state?
22) Process B is in which state?
23) Process C is in which state?

Step 5: The time-slice of the running process expires. Process C is scheduled.

24) Process A is in which state?
25) Process B is in which state?
26) Process C is in which state?

Step 6: The previously issued I/O request completes; the process that issued that I/O request is scheduled.

27) Process A is in which state?
28) Process B is in which state?
29) Process C is in which state?

Part 3. CPU Job Scheduling [2 points each]

Assume a workload with the following characteristics:

Job Name	Arrival Time (seconds)	CPU Burst Time (seconds)
A	0	8
B	2	4
C	5	7

If needed, assume a time-slice of 1 sec.

30) Given a FIFO scheduler, what is the **turnaround time** of job B?
   a. 4 seconds
   b. 10 seconds
   c. 12 seconds
   d. 14 seconds
   e. None of the above

31) Given a FIFO scheduler, what is the **average turnaround time** of the three jobs?
   a. 10 seconds
   b. 10 1/3 seconds
   c. 10 2/3 seconds
   d. 13 seconds
   e. None of the above

32) Given an SJF scheduler, what is the **turnaround time** of job C?
   a. 12 seconds
   b. 14 seconds
   c. 15 seconds
   d. 19 seconds
   e. None of the above

33) Given an SJF scheduler, what is the **average turnaround time** of the three jobs?
   a. 7 seconds
   b. 8 seconds
   c. 9 seconds
   d. 13 seconds
   e. None of the above

34) Given an RR scheduler, what is the **turnaround time** of job B?
   a. 8 seconds
   b. 9 seconds
   c. 10 seconds
   d. 11 seconds
   e. None of the above

35) Given an STCF scheduler, what is **the average turnaround time** of the three jobs?
   a. 10 seconds
   b. 10 1/3 seconds
   c. 10 2/3 seconds
   d. 24 1/3 seconds
   e. None of the above

36) Assume an STCF scheduler for the original workload with jobs A, B, and C. Assume a new job **D requiring 5 seconds of CPU** time arrives at some time T. For which arrival times of T would D preempt the job running at that time?
   a. 3 seconds
   b. 5 seconds
   c. 8 seconds
   d. 13 seconds
   e. None of the above **OR More than one of the above**

37) Assume an STCF scheduler for the original workload with jobs A, B, and C. Assume a new job **E arrives at time 8.5 seconds.** What is the longest that its CPU burst could be to preempt the job that was running at time 8.5? Pick the best answer.
   a. 1 second
   b. 2 seconds
   c. 3 seconds
   d. 4 seconds
   e. None of the above

Part 4: Virtualizing Memory [3 points each]

Designate if the statement is True (a) or False (b).

38) The **heap** and **stack** are statically allocated portions of a process' address space.

39) The **address space** for a process contains all of physical memory.

40) With **dynamic relocation**, the OS determines where the address space of a process is allocated in physical memory.

41) Two different address spaces (that do not share any pages) must contain different **valid vpn**'s from one another.

42) An **MMU** converts physical addresses to logical addresses for user-level processes.

43) With **segmentation**, the address space of each process must be allocated **contiguously** in physical memory.

44) With segmentation, each segment has its own **base** and its own **bounds**.

45) With pure segmentation (and no other support), fetching and executing an instruction that performs a store from a register to memory will involve exactly **four memory references**.

46) The **number** of virtual pages can be different than the number of physical pages.

47) If **10 bits** are used in a virtual address to designate an offset within a page, each page must be exactly **1 KB**.

48) If a physical address is 28 bits and each page is 4KB, the top 14 bits exactly designate the physical page number.

49) If a virtual address is 8 bits and each page is 32 bytes, then each address space can contain up to 8 pages.

50) Given a constant number of bits in a virtual address, the size of a linear page table decreases with larger pages.

51) Given a fixed page size, the size of a linear page table decreases with a smaller address space.

52) Given a 20-bit virtual address and 1KB pages, each linear page table will consume 1KB.

53) Compared to pure segmentation, a linear page table doubles the number of memory references (assuming no TLB).

54) A workload that sequentially accesses data has good **temporal locality**.

55) On a **TLB miss**, a new page in physical memory is allocated by the OS.

56) A **page fault** occurs if the **valid bit** is clear (equals 0) in a PTE needed for a memory access.

57) With paging, a single physical page can be **shared** across two address spaces only if they have the same virtual page number in each address space.

58) A multi-level page table typically reduces the amount of memory needed to store page tables, compared to a linear page table.

59) Given a 2-level page table (and no TLB), exactly 3 memory accesses are needed to fetch an instruction.

60) In the **memory hierarchy**, a backing store is larger than the memory layer above that uses that backing store.

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

62) If a clean page and a dirty page have both been accessed recently, the page replacement algorithm should replace the **clean** page over the **dirty** page.

63) A TLB miss is usually slower to handle than a page miss.

64) A different user-level process should be scheduled when a page miss is being handled.

65) LRU always performs as well or better than FIFO.

66) LRU with N+1 pages of memory always performs better than LRU with N pages of memory.

67) FIFO with N+1 pages of memory always performs better than FIFO with N pages of memory.

Part 5. Dynamic Relocation [2 points each]

Assume you have an architecture with 1KB address spaces and 16KB of physical memory. Assume you are performing **dynamic relocation with a base-and-bounds register**. The base register contains **0x00001acf (decimal 6863)** and the bounds register contains **292 (decimal)**. Translate each of the following virtual addresses into physical addresses.

68) Virtual address 0x0000019e (decimal: 414) is physical address:
   a. 0x000002c2 (decimal: 706)
   b. 0x00001c6d (decimal: 7277)
   c. 0x00007277 (decimal: 29303)
   d. Segmentation Violation
   e. None of the above

69) Virtual address 0x0000007d (decimal: 125) is physical address
   a. 0x000001a1 (decimal: 417)
   b. 0x00001b4c (decimal: 6988)
   c. 0x00006988 (decimal: 27016)
   d. Segmentation Violation
   e. None of the above

70) Virtual address 0x000000ee (decimal: 238) is physical address
   a. 0x00000c05 (decimal: 3077)
   b. 0x0000051d (decimal: 1309)
   c. 0x0000051e (decimal: 1310)
   d. Segmentation Violation
   e. None of the above

Part 6. Reverse Engineering the Page Table [2 points each]

Assume dynamic relocation is performed with a **linear page table**. Assume a system with the following parameters:

- address space size is 32KB
- physical memory size is 128KB
- page size is 4KB

Assume you are given the following trace of virtual addresses and the physical addresses they translate to. Can you reverse engineer the contents of the page table for this process?

```
 VA 0x00006e19 --> 0003e19
 VA 0x00004d35  --> 0000ad35
 VA 0x000030d8 --> 000050d8
 VA 0x0000244d --> 0001a44d
 VA 0x00005665 -->  Invalid
 VA 0x00000084 --> 0000d084
```

71) Page Table Entry 0
   a. Valid, PFN = 0x00
   b. Valid, PFN = 0xd0
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

72) Page Table Entry 1
   a. Valid, PFN = 0x01
   b. Valid, PFN = 0x10
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

73) Page Table Entry 2
   a. Valid, PFN = 0x01
   b. Valid, PFN = 0x1a
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

74) Page Table Entry 3
   a. Valid, PFN = 0x06
   b. Valid, PFN = 0x6e
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

75) Page Table Entry 4
   a. Valid, PFN = 0x0a
   b. Valid, PFN = 0xad
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

76) Page Table Entry 5
   a. Valid, PFN = 0x03
   b. Valid, PFN = 0x30
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

77) Page Table Entry 6
   a. Valid, PFN = 0x03
   b. Valid, PFN = 0x30
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

78) Page Table Entry 7
   a. Valid, PFN = 0xd0
   b. Valid, PFN = 0x1a
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

79) Page Table Entry 8
   a. Valid, PFN = 0x03
   b. Valid, PFN = 0x1a
   c. Invalid Entry or page table does not contain entry for this VPN
   d. Contents of PTE cannot be determined from this address trace
   e. None of the above (e.g., Valid, but a different PFN)

Part 8. Hitting or Missing in the TLB [2 points each]

The following questions ask you to calculate the miss rate (or hit rate) for the TLB. Assume you have a virtual address that requires 16 bits and there are 512 possible virtual pages per address space. You should ignore all instruction references (i.e., do not consider how they impact the contents of the TLB). Assume the array is page-aligned.

80) Assume you have a **1-entry TLB**. Assume the running process sequentially accesses contiguous **4-byte integers** in an extremely large array, starting at index 0. What will be the TLB miss rate?

a) 1/1
b) ¼
c) 1/32
d) 1/128
e) None of the above or Not enough information to answer

81) Assume you have a **4-entry TLB** with LRU replacement for the same workload as above. What will be the miss rate in the TLB?

a) 1/1
b) ¼
c) 1/8
d) 1/32
e) None of the above or Not enough information to answer

82) Assume you have a **1-entry TLB** and the running process accesses every **fourth element** (i.e., every fourth 4-byte integer) in the array (i.e., index 0, then index 4, then index 8, etc…). What is the miss rate for this access pattern?

a) ¼
b) 1/8
c) 1/32
d) 1/64
e) None of the above or Not enough information to answer

83) Assume the running process **repeatedly** accesses every 4-byte integer in a **128 integer array** in a loop (i.e., accesses elements 0 through 127, then elements 0 through 127 again, over and over). To have a **hit rate that approaches 1/1**, at least how many entries must be in the TLB? Assume the TLB uses LRU.

a) 1
b) 2
c) 3
d) 4
e) None of the above or Not enough information to answer

84) If the TLB contains **1 less entry** than the number of entries you calculated for the previous question, what will be the TLB hit rate? Assume the TLB uses LRU.

a) 4/5
b) 3/4
c) 2/3
d) 0
e) None of the above or Not enough information to answer

85) Assume the running process repeatedly accesses the first 4-byte integer on each page of an array that fits on 8 pages (i.e., in a loop). Assume the TLB has exactly 8 entriess, the TLB uses LRU, and the TLB does not support ASIDs. Assume the OS performs a context-switch to another process that runs for only one memory reference (i.e., one address translation) and then switches back to the original process. What will be the hit rate for the **next iteration t**hrough the loop for the original process?

a) 7/8
b) 3/4
c) 1/2
d) 0
e) None of the above or Not enough information to answer

Part 7. Multi-level Page Tables [2 points each]
Assume dynamic relocation is performed with a **two-level page table** with no TLB. Assume the page size is an unrealistically-small 32 bytes, the virtual address space for the current process is 1024 pages, or 32 KB, and physical memory consists of 128 pages. Thus, a virtual address needs 15 bits (5 for the offset, 10 for the VPN) and a physical address requires 12 bits (5 offset, 7 for the PFN). The upper five 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 32 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 an 8-bit PTE is VALID | PFN6 ... PFN0.

You also know that the PDBR points to page 51 (decimal) and the contents of memory are as follows:

```
page   0: 7f 7f 7f 7f 7f 7f 7f 7f e4 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page   1: 7f e5 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ec 7f 7f 7f 7f 7f 7f 7f 7f 7f
page   2: 7f ca 7f 7f e3 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ce
page   3: 07 19 19 19 0e 06 1a 0d 1e 01 03 0d 19 10 08 0f 09 1d 0c 07 12 0d 1a 01 1b 08 0c 1a 02 0e 12 10
page   4: 0e 03 17 06 05 0d 02 1c 1a 02 07 19 17 10 14 0d 12 13 17 1e 10 04 17 1e 10 1c 17 17 18 12 03 04
page   5: 05 1c 0c 01 06 03 0f 07 1d 0a 07 11 19 10 09 14 1e 10 19 13 11 08 11 0d 0b 10 16 0c 18 00 1a 11
page   6: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f bf dc 7f 7f 7f 7f 7f 7f e6 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page   7: 1b 03 1c 0d 10 17 1a 03 12 05 1d 17 07 0a 1e 1d 17 01 06 12 06 10 00 0e 0e 19 0a 1b 07 00 0e 05
page   8: 0c 15 10 03 1b 18 1c 11 01 00 10 0b 1b 01 12 0d 1d 14 09 1e 14 1d 1e 1b 00 0c 16 07 02 1b 1e 04
page   9: 19 06 0d 0d 16 11 16 1d 14 00 1e 0b 0c 09 1a 0b 01 0d 06 0d 00 18 1b 13 0c 04 06 13 01 0d 1b 1e
page  10: 12 12 1c 14 05 10 1d 19 16 07 0d 12 03 0c 0d 0b 17 08 0f 1d 12 10 16 15 11 0b 11 1c 0c 14 08 10
page  11: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 9a 7f 7f 7f 7f 7f 7f 7f 7f 7f c5 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  12: 05 05 04 06 11 17 18 1a 03 19 05 02 08 12 01 06 0c 19 0c 0c 0b 02 1b 1c 16 0c 0d 09 14 14 09 1b
page  13: 0f 1c 0e 04 04 1d 15 0a 0f 1b 0d 06 19 12 0f 05 13 10 08 1e 0f 13 12 05 01 1b 0a 0d 06 12 17 0b
page  14: 00 19 09 0a 01 14 07 10 0a 0a 01 0a 1c 1d 1c 1a 19 13 12 16 1d 12 11 16 1a 10 00 12 0a 03 01 10
page  15: 07 1b 1a 1b 0a 1a 0b 10 13 09 07 18 18 09 08 1e 0d 19 0b 0a 05 1d 17 0b 12 01 0c 0c 09 1b 16 12
page  16: 14 1e 04 1e 14 17 1d 10 10 04 0c 11 08 0d 0b 19 0a 1b 07 14 0f 09 18 1e 03 19 02 0b 1d 1d 1c 0b
page  17: 07 0f 08 02 14 0f 1e 03 17 00 15 0c 06 03 02 10 13 02 11 08 19 08 12 10 11 11 19 00 09 01 01 0e
page  18: 7f 90 8c 7f 7f 7f c3 7f 7f a8 84 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f c8 7f 8d
page  19: 1b 09 06 05 15 13 1d 1a 0e 14 00 19 03 19 11 1b 17 0a 0d 16 05 00 1d 02 19 05 10 14 0b 09 11 14
page  20: 15 14 13 09 03 13 17 11 1e 1b 0c 10 17 07 00 08 1a 16 07 04 0b 19 1c 0b 19 0e 10 03 08 04 00 0d
page  21: 00 0b 14 1b 12 02 06 0f 07 04 04 18 0a 1a 1d 0a 14 06 08 06 0f 03 19 15 03 08 07 1d 1b 06 13 17
page  22: 17 07 0b 15 10 1d 0f 0d 13 06 1b 0b 0c 15 04 14 1a 0a 1d 10 18 0e 0c 08 00 06 1e 10 0e 1b 12 1e
page  23: 01 04 1d 12 1b 02 06 16 05 04 0a 06 0f 0e 0b 10 07 15 1b 12 10 09 1b 1b 10 06 07 0a 14 1d 11 09
page  24: 7f fb 7f 7f 7f 7f 7f 89 7f 7f 7f 7f 7f 88 7f 7f 7f f8 7f fa 7f 7f 7f 7f 7f 7f 7f 7f 7f be 7f 7f
page  25: 7f 87 9e 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  26: 04 03 12 03 1d 09 13 01 11 0a 1d 14 14 04 05 18 11 17 02 14 13 18 15 07 09 10 1c 11 07 1d 01 15
page  27: 04 18 12 1e 1d 14 01 00 0e 19 02 0d 02 09 00 0a 10 06 18 01 06 07 0c 17 0a 15 1c 04 1a 12 17 12
page  28: 14 12 14 07 1c 12 03 06 1a 0e 0f 0a 0f 08 1e 10 14 07 06 1b 07 12 0d 0f 0c 1d 00 03 07 11 15 0f
page  29: 19 0b 0b 09 0b 10 18 00 05 06 13 17 02 00 03 0d 15 1e 0b 0d 1b 17 1c 08 16 19 08 07 1c 06 0a 03
page  30: 07 0d 03 12 19 05 13 15 09 0b 04 13 1e 18 06 0d 0f 08 1d 1b 0d 07 17 10 16 1b 1e 18 06 11 16 05
page  31: 7f 7f 7f 7f 7f bd 7f 7f 7f 7f 7f 7f 96 7f 7f 7f 7f 7f 7f 7f de 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f3
page  32: 1b 1b 0a 09 1d 1b 05 03 0c 18 0c 08 00 0a 01 09 00 05 0f 0b 1b 0f 05 05 0b 19 05 1c 0f 04 10 08
page  33: 06 14 16 07 15 0c 00 0c 01 06 06 0b 17 0d 19 1b 1b 19 0f 0c 18 01 17 1a 00 04 1e 11 18 0a 1b 01
page  34: 1a 00 02 00 02 02 02 16 1c 18 19 18 05 0b 01 05 06 0e 1c 0a 15 14 09 01 06 06 0f 09 14 12 03 1a
page  35: 7f 7f 7f 7f 7f 7f 7f 7f 7f db 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f bb
page  36: 11 18 11 18 09 0b 14 1a 1c 16 05 12 02 05 0d 13 06 0d 09 03 1b 19 18 04 19 1c 02 07 09 06 11 16
page  37: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 95 7f 7f 7f 7f 7f 7f
page  38: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f b2 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 94 7f 7f
page  39: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f a2 7f 7f 7f 7f b7 7f 7f e2 7f 7f 7f
page  40: 14 06 18 1c 05 0e 15 14 01 01 09 05 1d 11 1e 08 0c 1a 13 11 06 03 1e 07 15 14 0d 14 1b 02 17 12
page  41: 17 0d 07 15 1a 0d 0b 08 10 04 02 14 04 15 12 17 0a 1c 01 02 10 07 01 0e 1b 0c 0d 19 1a 05 15 14
page  42: 11 10 13 12 0c 05 12 01 01 0a 07 17 04 0d 15 0c 18 03 05 08 0b 11 06 11 15 14 19 02 15 16 19 04
page  43: 02 0b 08 06 1d 0b 1b 1a 04 1c 04 1d 07 1d 19 08 13 06 12 06 04 07 18 13 11 08 16 1e 0d 0e 12 16
page  44: 19 03 07 19 07 0c 12 1c 1c 0c 0f 10 09 15 01 14 01 09 09 17 06 0c 0e 01 14 05 02 1d 1c 0f 07 0d
page  45: 19 06 00 06 0c 05 15 13 06 08 08 1a 11 0b 14 1d 16 06 09 0c 0d 16 02 0b 1b 1d 1b 19 15 13 05 19
page  46: 09 16 08 1a 13 1e 08 0b 0a 12 0c 1e 02 1e 05 17 01 10 10 16 1b 07 1c 0a 06 18 07 13 0b 11 07 07
page  47: 7f 7f 7f 7f 7f d9 7f 7f 7f 7f 7f 7f 91 7f 7f 7f 7f 7f e8 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  48: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  49: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  50: 04 1d 1b 19 09 03 0d 02 0a 12 09 0a 0a 19 01 19 12 10 15 00 1a 08 0c 0f 1b 07 10 1e 06 14 05 11
page  51: cd a6 d2 7f a3 d4 9f c4 c0 98 99 f5 af 92 80 dd 81 cb b9 7f 86 82 d7 d6 c9 c6 ef a5 b8 8b ea a7
page  52: 13 02 0c 15 03 07 16 19 15 15 0b 19 0b 0a 06 0f 0a 0b 16 1b 09 05 1e 04 13 04 0d 1e 14 17 16 16
page  53: 16 1c 15 1b 0b 09 11 11 0a 0c 0e 01 1b 16 13 03 0b 0b 10 10 1a 0e 05 16 1c 1d 15 04 01 08 15 13
page  54: 13 0d 1e 1d 12 0b 08 08 19 1a 16 0a 16 10 05 08 18 0c 17 13 1c 11 10 12 16 06 1d 12 10 07 08 14
page  55: 10 00 01 1e 08 10 1e 13 05 1c 0f 1b 1c 10 1a 16 03 02 09 1a 02 1b 09 15 16 0a 18 10 0d 1a 16 19
page  56: 7f cf 7f ad 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f d8 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  57: 7f 7f 7f 7f 7f 7f 7f ab 7f ac 7f 7f 8f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f2 7f 7f 7f 7f 7f 7f 7f
page  58: 1c 0c 00 1d 06 0e 12 1b 0b 0d 06 1d 1d 14 00 0e 17 16 01 13 05 09 0a 0d 11 09 1b 02 19 17 1a 14
page  59: 1a 13 16 05 11 15 01 0c 14 04 1d 06 03 15 1e 17 10 0a 06 01 06 13 07 1e 04 07 0c 1d 0a 05 16 02
page  60: 16 04 1c 17 0a 1a 07 03 09 0a 02 1a 1b 1e 08 13 0e 0d 12 17 06 10 1b 15 1d 01 0a 05 07 0e 10 12
page  61: 11 16 01 12 02 10 06 16 0e 03 0f 0e 16 1b 1e 1e 01 10 0c 19 07 1c 04 04 07 0a 19 0b 11 1a 02 0f
page  62: 06 09 0d 14 09 02 0e 0d 0e 0b 0d 0d 09 0e 11 05 0e 19 0f 0a 01 0b 13 0b 1e 1c 04 15 05 00 1d 0e
page  63: 01 0d 12 05 04 14 15 10 0b 11 04 07 03 0d 0c 11 14 0b 01 16 16 1d 1e 0c 0f 1c 06 04 0f 12 08 05
page  64: 7f 7f 7f 7f f1 7f d3 7f 7f 7f ed 7f 7f 7f 7f 7f 7f 7f 7f ae 7f 7f 7f 7f c2 7f 7f 7f 7f 7f 7f 7f
page  65: 07 17 1d 07 02 0b 16 0b 12 10 17 1c 05 0c 0b 05 09 08 0d 1e 11 0f 0d 05 14 1d 14 0d 19 06 0a 08
page  66: 17 17 04 0e 0d 09 12 05 17 01 1e 14 16 17 0c 0f 15 1c 1b 0e 0f 05 17 0d 0c 06 14 0e 03 07 0b 12
page  67: 0b 0c 04 19 03 12 01 1d 0d 09 15 0b 01 07 07 00 1e 18 1b 1a 0d 0d 06 19 0c 08 0e 18 06 1e 0c 10
page  68: 7f 7f 93 7f 7f 7f 7f 7f 7f 7f 7f 7f f4 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  69: 0a 18 16 09 08 10 02 04 0c 1e 1d 01 16 14 13 1d 1e 10 14 1a 04 0e 1c 00 0b 09 05 0d 0b 07 1d 1b
page  70: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f6 7f 7f a1 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  71: 05 0a 1d 15 16 09 14 0a 06 04 05 02 0c 1a 10 0b 13 1c 08 1c 1e 0a 01 15 1c 0b 09 07 03 14 08 1c
page  72: 00 06 1b 05 09 1e 07 11 0d 00 13 0f 1d 1e 02 12 0a 04 1c 02 0f 07 11 06 1a 0d 06 18 04 16 07 1a
page  73: 7f 7f 7f 7f 7f 7f a0 7f 7f 7f 7f b5 83 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  74: 18 06 14 1e 00 0d 1d 08 19 19 15 1e 15 03 1a 17 0b 02 08 10 07 1e 04 08 03 17 19 07 0c 1b 12 06
page  75: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f9 7f 7f 7f 7f 7f
page  76: 12 0d 03 10 12 12 1b 11 0b 11 16 0c 19 16 18 01 13 0b 12 01 0c 0f 12 09 00 00 16 19 19 0d 0b 1a
page  77: 7f 8a 7f 7f 7f 9c 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ff eb 97 7f 7f 7f 7f 7f 9d 7f 7f 7f 7f
page  78: 03 15 15 0b 08 10 0d 0c 0e 1c 0b 00 00 0b 05 18 1c 0d 1b 11 1d 0e 1a 1b 03 10 06 18 13 09 14 1e
page  79: 0d 00 17 02 0b 16 08 17 1b 15 0d 1c 09 1e 12 10 1b 03 08 18 02 1c 0e 0f 1e 02 00 11 13 1a 05 1b
page  80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  81: 17 1a 07 1a 0a 0c 03 10 00 09 14 17 05 18 0d 06 17 16 0e 10 13 13 17 17 06 00 03 09 11 1d 0c 0b
page  82: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f e1 a9 ba 8e 7f 7f 7f 7f
page  83: 1a 10 07 10 15 0d 10 02 07 17 0a 05 18 00 01 01 05 01 09 08 1c 07 11 09 16 03 0a 04 09 08 0e 1d
page  84: 7f 7f 7f 7f 7f a4 7f 7f 7f 7f 7f 7f 7f 7f 7f d1 cc 7f 7f 7f 7f 7f 7f 7f 7f b4 7f 7f 7f 7f 7f 7f
page  85: 1d 0d 1d 09 08 0f 1c 1b 08 1c 04 0a 10 00 19 17 17 18 18 11 1d 19 09 0a 02 1a 03 04 13 13 0f 0a
page  86: 7f 85 e7 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f fd 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  87: 7f 7f 7f c1 7f b6 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page  88: 00 0a 19 10 06 0c 14 05 07 1a 0c 0c 1e 1d 09 00 05 1e 15 07 1c 1b 16 00 10 06 07 0c 0c 1b 0b 05
page  89: 1a 00 1c 1b 0c 15 00 09 16 1d 09 06 12 15 0e 0f 08 1b 0b 0f 02 02 0d 00 12 18 1c 0c 07 05 1e 0e
page  90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  91: 10 15 02 1e 15 10 11 0a 0b 0e 17 04 00 19 19 02 12 0b 0b 15 10 08 1e 14 06 14 01 0c 05 02 13 19
page  92: 0b 14 03 00 18 04 0f 0a 0d 1a 18 16 1a 1a 17 02 11 05 1b 17 19 10 0c 0a 1d 1b 04 02 14 08 10 13
page  93: 7f 7f fc 7f 7f 7f 7f 7f 7f 7f e0 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f aa 7f 7f 7f 7f 7f 7f
page  94: 0a 0f 0a 19 15 1c 0f 0e 09 08 1b 0b 1b 1a 1c 11 17 1e 0e 0e 1e 04 0d 17 1d 04 0a 0c 01 00 06 13
page  95: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page  96: 0c 0c 19 17 00 09 09 08 03 0e 0e 13 0c 18 16 1c 00 19 1e 0f 10 1c 09 19 0e 1a 16 0e 04 08 13 0a
page  97: 05 1a 16 13 17 12 0a 01 0a 13 05 05 03 0f 1d 16 00 0b 15 03 18 07 0d 18 1b 02 19 1b 19 17 0b 09
page  98: 0d 0f 13 0e 04 0a 03 0f 13 12 02 11 18 11 0a 18 0c 18 00 02 0e 02 06 1c 18 09 03 16 15 05 11 13
page  99: 12 0c 17 0d 0e 0b 0f 0f 07 15 1c 00 1e 19 1e 0c 05 0f 1c 06 19 17 11 14 1d 11 1a 1c 14 11 1c 06
page 100: 12 0a 07 07 09 02 03 01 0a 1c 0a 1a 05 16 1d 06 12 16 00 00 0d 08 0b 10 18 07 1a 08 14 1b 03 1d
page 101: 07 04 1c 1a 18 15 0f 18 1b 1b 03 07 08 13 00 19 1b 07 07 19 0e 06 0e 03 16 10 0d 1c 1a 06 0b 0e
page 102: 01 0a 06 07 01 03 0e 1e 0d 01 1a 11 11 0d 00 02 0b 1c 00 0c 01 15 05 07 02 00 0b 13 04 1d 1c 1b
page 103: 07 09 17 02 0d 1d 07 0f 1c 1b 18 10 1c 07 1b 0c 14 12 08 15 12 1e 14 0d 0b 1b 0a 14 15 1d 05 14
page 104: 1c 01 00 04 0f 16 03 03 01 0b 0d 16 10 18 10 07 08 11 05 13 11 17 0d 1d 1d 15 05 1e 12 04 08 18
page 105: 17 00 0a 03 06 0b 09 1c 1b 13 0a 0b 1b 02 0c 02 13 0f 11 03 0a 05 1d 0e 02 05 0e 09 12 1c 0d 12
page 106: 7f 7f d5 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f c7 7f 7f 7f 7f 7f f0 7f 7f 7f 7f 7f 7f 7f 7f
page 107: 0f 18 1a 1b 08 1c 18 11 05 19 18 1d 15 0f 09 11 0d 1b 0a 10 16 1d 0e 03 1e 10 01 0f 15 15 0c 01
page 108: 1d 17 13 08 02 0c 0f 11 05 0b 04 00 0f 0e 12 0d 14 1d 0c 12 10 0f 00 02 11 12 0b 0a 03 15 13 03
page 109: 12 1e 0f 1c 05 1b 10 03 12 09 00 02 00 0d 19 12 14 03 13 0d 03 0e 19 14 18 00 08 1e 01 05 0c 02
page 110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 111: 7f 7f 7f 7f 7f 7f 7f 7f 7f f7 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f bc 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 112: 09 11 05 16 06 15 18 12 08 19 1d 00 18 18 18 10 04 0b 15 14 02 00 03 18 00 02 02 01 03 18 1e 19
page 113: 1b 1d 1c 1a 16 1d 06 04 19 1d 10 10 13 15 0d 03 05 0d 03 01 12 1d 0c 18 11 0d 11 1e 03 11 08 12
page 114: 17 04 0e 07 0b 0b 02 0f 06 10 10 0d 1b 10 14 1d 0d 1a 0c 00 06 07 14 05 14 09 14 1c 14 15 12 11
page 115: 13 14 17 1b 03 07 00 0a 14 0b 12 1e 0d 12 10 0e 03 0c 18 17 1b 1b 1a 0c 1e 0a 0a 05 07 15 18 01
page 116: 02 0e 18 15 03 0c 09 06 07 19 11 18 0c 1a 00 0a 05 08 16 05 1b 11 1b 00 01 01 1e 0e 01 08 0a 08
page 117: 7f e9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 9b 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 118: 03 10 09 12 00 01 0b 08 17 02 04 14 15 1d 15 06 08 17 0f 00 11 0c 15 00 11 1c 10 0c 05 04 0d 0c
page 119: 1c 08 19 0b 05 14 0a 09 13 14 00 00 10 02 03 1c 1d 16 1c 15 1c 00 0b 0b 0a 08 09 17 1d 1c 0c 12
page 120: 08 08 0f 13 12 0e 0e 1e 1e 03 0f 06 05 0f 06 0e 1c 13 0f 14 1c 13 0b 07 12 07 1b 0c 16 09 1a 1d
page 121: 09 17 1e 18 0c 05 0e 03 04 0d 1a 15 0c 1d 05 07 08 11 1b 0b 19 1d 0e 1b 1b 0e 05 02 07 0e 00 0a
page 122: 0d 1a 03 05 02 0e 0e 01 11 12 15 12 01 01 0f 07 09 15 15 0a 19 03 03 05 1b 11 14 00 11 1a 0f 16
page 123: 0d 1d 06 04 17 11 03 0f 07 09 1d 1e 16 05 07 17 10 0e 09 1d 07 0c 03 1c 14 04 18 1d 0e 15 10 18
page 124: 0d 0f 1d 14 1e 0d 1b 0d 08 1d 13 04 11 15 04 0e 0d 05 15 0f 12 10 13 18 0f 1d 1b 0e 03 0f 0b 1b
page 125: 02 19 16 1c 0d 13 17 0f 16 06 01 06 1e 0d 1b 1e 1c 11 08 12 17 16 01 01 01 07 0c 0b 17 17 08 0a
page 126: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 127: 0b 1e 0b 05 12 07 0a 14 15 13 08 05 04 03 1d 0a 0c 04 1a 03 13 04 17 1d 13 04 08 1d 08 02 13 07
```

86) When accessing **virtual address 0x5c5b**, what will be the first page accessed (decimal)?

a. 5
b. 51
c. 54
d. 64
e. Error or None of the above

87) What **index** of the page directory will be accessed first (hexadecimal)?

a. 0x05
b. 0x17
c. 0x5c
d. 0xb1
e. Error or None of the above

88) What will be the **second page** accessed?

a. 0x64 (decimal: 100)
b. 0x66 (decimal: 102)
c. 0x54 (decimal: 84)
d. 0x56 (decimal: 86)
e. Error or None of the above

89) What are the **contents** of the corresponding PTE that will be read?

a. 0x67
b. 0xd7
c. 0xe7
d. 0xf7
e. Error or None of the above

90) What is the **final physical address** for this virtual address?

a. 0x5fb
b. 0xc5b
c. 0xcfb
d. 0x51fb
e. Error or None of the above

91) What are the **contents** (i.e., the value) at that physical address?

a. 0x14
b. 0x5b
c. 0x84
d. 0x514
e. Error or None of the above

92) When accessing **virtual address 0x0fa3**, what are the **contents** at the corresponding physical address?

a. 0x04
b. 0x14
c. 0x5b
d. 0x84
e. Error or None of the above

Congratulations on finishing your first CS 537 Exam!