CS 537: Introduction to Operating Systems
Fall 2015: Midterm Exam #1

This exam is closed book, closed notes. All cell phones must be turned off. No calculators may be used.

You have two hours to complete this exam.

There are two parts to this exam: the first is true/false, the second is multiple choice. Some of the T/F questions are very simple, and some will take you awhile to determine. We expect you'll end up spending approximately equal amounts of time on each part.

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).

Good luck!

Part 1: Straight-forward True/False [1 point each]
Designate if the statement is True (a) or False (b).

1) An **operating system** is defined as hardware that converts software into a useful form for applications.

   False. It is software that converts hardware.

2) Examples of **resources** that the OS must manage include CPU, memory, and disk.

   True.

3) The **abstraction** that the OS provides for the CPU is a virtual address space.

   False. The CPU abstraction is a process.

4) A **process** is defined as an execution stream (or thread of control) in the context of a process state.

   True.

5) The **address space** of a process is part of its process state.

   True.

6) A process is identical to a **program**.

   False; process is dynamic, program is static.

7) A process is identical to a **thread**.

   False; can have multiple threads in a single process.

8) Two processes reading from the **same virtual address** will access the same contents.

   False; each process has its own address space.

9) A modern OS virtualizes a single CPU with **time-sharing**.

   True.

10) Entering a **system call** involves changing from **user mode** to **kernel mode**.

   True.

11) When a user-level process wishes to call a function inside the kernel, it directly jumps to the desired function.

   False; must use system call; after generating a trap which is handled by the OS with a trap handler, the desired system call is invoked through the system call table.

12) An example of a **mechanism** inside the OS is the process **dispatcher**.

   True.

13) **Cooperative multi-tasking** requires hardware support for a timer interrupt.

   False; cooperative assumes process will voluntarily relinquish CPU or enter OS.

14) A timer tick is identical to a **time slice.**

   False; a time slice (how long a process is scheduled with RR) may be multiple timer ticks long.

15) PCB stands for **process control base**.

   False; Process Control Block.

16) On a uniprocessor system, there may only be one **ready** process at any point in time.

   False; just one RUNNING process, but many could be ready and want to be scheduled.

17) A **FIFO scheduler** schedules ready processes according to their arrival time.

   True.

18) The **convoy effect** occurs when high priority jobs must wait for lower priority jobs.

True/False both accepted; convoy effect usually implies short jobs must wait for long jobs, but one could consider a short job to be high priority and a long job to be low priority.

19) A **SJF scheduler** uses the past run time (i.e., cpu burst) of a job to predict future run time (i.e., cpu burst).

False; SJF assumes an oracle with perfect knowledge of future behavior.

20) A **STCF scheduler** guarantees that it will schedule the ready job with the smallest remaining cpu burst.

True.

21) A **RR scheduler** may preempt a previously running job.

True.

22) An RR scheduler tends to decrease average response time as the time-slice is decreased.

True; jobs will be scheduled sooner (decreasing response time) with a smaller time slice.

23) The shorter the time slice, the more a RR scheduler gives similar results to a FIFO scheduler.

False; with a longer time slice (imagine the time slice > cpu burst), RR schedules the jobs without rotating between jobs.

24) If all jobs arrive at the same point in time, a SJF and an STCF scheduler will behave the same.

True.

25) If all jobs have identical run lengths, a FIFO and a SJF scheduler will behave the same.

True.

26) If all jobs have identical run lengths, a RR scheduler provides better average turnaround time than FIFO.

False; if all jobs are identical, RR is horrible for turnaround time because all jobs will complete at nearly the same time.

27) A RR scheduler is guaranteed to provide the optimal average turnaround time for a workload.

False; RR can provide horrible turnaround time (see example in 26).

28) A SJF scheduler requires an **oracle** to predict how long each job will perform I/O in the future.

False; needs a oracle, but it is to predict the next CPU burst of each job.

29) With a **MLFQ scheduler,** compute-bound jobs are given higher priority.

False; jobs that do a lot of computation (long CPU burst) are given low priority.

30) With a MLFQ scheduler, high priority jobs have longer time-slices than low priority jobs.

False; high priorities have short time-slices so that interactive jobs can run promptly, but for just a short time.

31) With a MLFQ scheduler, jobs run to completion as long as there is not a higher priority job.

False; at the lowest priority level, MLFQ will still RR across jobs of equal priority.

32) The OS provides the illusion to each process that it has its own address space.

True.

33) The **static** portion of an address space cannot contain any data.

False; read-only data could be in a static portion (along with code).

34) **Stacks** are used for procedure call frames, which include local variables and parameters.

True.

35) Pointers should not reference the **heap.**

False; it can be bad practice to return pointers to data allocated on the stack.

36) An **instruction pointer register** is identical to a program counter.

True.

37) A modern OS virtualizes memory with **time-sharing**.

False, use space-sharing.

38) A **virtual address** is identical to a **logical address**.

True.

39) With **dynamic relocation**, hardware dynamically translates an address on every memory access.

True.

40) An **MMU** is identical to a **Memory Management Unit**.

True.

41) The OS may not manipulate the contents of an MMU.

False; OS sets up contents of MMU when switch to new process.

42) A disadvantage of **segmentation** is that different portions of an address space cannot grow independently.

False; segmentation lets each segment grow independently.

43) A disadvantage of segmentation is that **segment tables** require a significant amount of space in memory.

False; segment tables are usually small (just a base and bounds for each segment and not many segments).

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

False; fetching the instruction requires one memory reference and executing the store requires a second memory reference.

45) Paging approaches suffer from **internal fragmentation**, which grows as the size of a page grows.

True.

46) A **physical page** is identical to a **frame**.

True.

47) The **size** of a virtual page is always identical to the size of a physical page.

True.

48) The **number** of virtual pages is always identical to the number of physical pages.

False; the size of the virtual address space can be different than the amount of physical memory.

49) If **8 bits** are used in a virtual address to designate an offset within a page, each page must be exactly **256 bytes**.

True; 2^8 = 256 bytes.

50) If a physical address is 24 bits and each page is 4KB, the top 10 bits exactly designate the physical page number.

False; lg 4K = 12 → 12 bits for page offset; 24 – 12 = 12 bits for physical page number.

51) If a virtual address is 16 bits and each page is 128B, then each address space can contain up to 512 pages.

True. Lg 128 = 7 bits. 16 – 7 = 9 bits. 2^9 = 512 pages.

52) A linear **page table** efficiently maps physical page numbers to virtual page numbers.

False; efficiently maps VPN to PPN (would be expensive to search other way through linear structure).

53) Given a fixed page size, the size of a linear page table increases with a larger address space.

True; larger address space $\rightarrow$ more pages $\rightarrow$ more page table entries in linear table.

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

False; larger pages $\rightarrow$ fewer pages $\rightarrow$ fewer entries in linear page table.

55) Given a 28-bit virtual address and 1KB pages, each linear page table will consume 2^{18} bytes.

False. 1KB pages -> 10 bit offsets. 28 – 10 = 18 bits for VPN. Each PTE is 4 bytes (coversheet assumption). 2^18 * 4 bytes.

56) Page table entries are stored in the PCB of a process when a context switch occurs.

False; keep base of page table in PCB, but not individual page table entries.

57) Compared to pure segmentation, a linear page table doubles the required number of memory references.

True; need to look up VPN->PPN in page table on every memory access.

58) A disadvantage of paging is that it is difficult to track **free** memory.

False; pages are all same size so just use bitmap to track state (free vs. allocated) of each page.

59) A disadvantage of paging is that all pages within an address space must be allocated.

False; each page table entry must be allocated (with a linear table), but if the page table entries shows the page isn't valid, then the page doesn't have to be allocated.

60) A **TLB** is identical to a **Translation Lookahead Buffer**.

True or False; question thrown out (exam worth only 194 points instead of 196). In class, we only used term Translation Lookaside Buffer, but both terms are sometimes used.

61) A TLB **caches** translations from full virtual addresses to full physical addresses.

False; TLB translates virtual page numbers to physical page numbers (no offset portion of address in TLB).

62) If a workload sequentially accesses 4096 4-byte integers stored on 256 byte pages, the TLB is likely to have a **miss rate** around 2^{-8} (ignore any other memory references).

False. Access 16KB of data sequentially; with 256 byte pages (and perfect alignment), this data fits on 2^14 / 2^8 = 2^6 pages. Assume first access to each page misses in TLB, while remaining accesses to that page hit in TLB. Miss rate = # misses / # accesses = # pages / # accesses = 2^6 / 2^12 = 2^(-6).

63) If a workload **sequentially** accesses data, the TLB miss rate will decrease as the page size increases.

True; with large pages, more accesses to same page, whose mapping will already be in TLB.

64) A workload that sequentially accesses data is likely to have good **temporal locality**, but not necessarily good **spatial locality**.

False; good spatial locality but not necessarily temporal.

65) **TLB reach** is defined as the number of TLB entries multiplied by the size of each TLB entry.

False; TLB entries multiplied by the size of each page.

66) On a **context switch**, the TLB must be flushed to ensure that one process cannot access the memory of another process.

False; can avoid flushing TLB if have ASIDs.

67) A longer scheduling time slice is likely to decrease the overall TLB miss rate in the system.

True; if a process is scheduled for a longer period of time, it will amortize the cost of the cold start misses to load up TLB with needed translations over a longer period of time.

68) On a **TLB miss**, the desired page must be fetched from disk.

False; TLB miss means the page tables must be accessed; page fault will need to access disk.

69) With a TLB, only the outermost page table of each process needs to be accessed.

False; if a TLB miss, still need to walk entire page table.

70) If the **valid bit** is clear (equals 0) in a PTE needed for a memory access, the running process is likely to be killed by the OS.

True; if process tries to access page not valid in its address space, it is a segmentation fault.

71) There is a separate page table for every active process in the system.

True.

72) An **inverted page table** is efficiently implemented in hardware.

False; inverted page tables are implemented in software.

73) One advantage of adding segmentation to paging is that it potentially reduces the size of the page table.

True; only need page table entries for valid pages in each separate segment.

74) One advantage of adding paging to segmentation is that it reduces the amount of internal fragmentation.

False; Paging has internal fragmentation.

75) A single page can be **shared** across two address spaces by having each process use the same page table.

False; if two processes use the same page table, all of their pages will be shared.

76) An advantage of a multi-level page table (compared to a linear page table) is that it potentially reduces the number of required memory accesses to translate an address.

False; multiple levels increase the number of memory accesses needed for translation.

77) A **page directory** is identical to the outermost level of the page table.

True.

78) With a multi-level page table, the complete VPN is used as an index into the page directory.

False (but question mistakenly scored as True); only outer bits of VPN are used. Question thrown out and thus exam is worth 194 points instead of 196.

79) TLBs are more beneficial with multi-level page tables than with linear (single-level) page tables.

True; if a TLB hit, able to avoid more memory lookups for page translation (higher cost of a miss).

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

False; 2 accesses for 2 levels of address translation + 1 for fetch = 3 accesses.

81) With a multi-level page table, hardware must understand the format of PTEs.

False; not necessary to have hardware support for multi-level page tables.

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

False; backing stores are larger and slower than the layers above that use it.

83) If the **present bit** is clear in a needed PTE, then the running process is likely to be killed by the OS.

False; if present bit is clear, page must be brought in from disk.

84) A **page fault** is identical to a **page miss**.

True.

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

False; dirty bit means contents of page in memory do not match contents on disk.

86) The OS can run a single process whose allocated address space exceeds the amount of physical memory available in the system.

True.

87) The OS can run multiple processes whose total allocated address space exceeds the amount of physical memory available in the system.

True.

88) When a page fault occurs, it is less expensive to replace a clean page than a dirty page.

True; clean page can be simply discarded since it matches what is on disk; dirty page must be written to disk to update that (only) copy.

89) When a page fault occurs, the **present** bit of the victim page (i.e., the page chosen for replacement) will be cleared by the OS.

True; the victim page will no longer be present in main memory.

90) A TLB miss is usually faster to handle than a page miss.

True; TLB miss just requires accessing main memory; page miss requires accessing much slower disk.

91) **Demand paging** is identical to **anticipatory paging**.

False; demand paging brings in page only when needed; anticipatory brings in pages beforehand.

92) **Prefetching** of pages helps sequential workloads to avoid page misses.

True; prefetching works well with sequential accesses since can predict next page to be accessed.

93) **LRU** is an example of a mechanism for determining which page should be replaced from memory.

False; LRU is a policy.

94) The **OPT** replacement policy replaces the page that is used the least often in the future.

False; OPT replaces the page that will be used the furthest away in the future (not least often).

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

False; not necessarily.

96) OPT always performs as well or better than FIFO.

True.

97) LRU with N+1 pages of memory always performs as well or better than LRU with N pages of memory.

True, due to stack property.

98) FIFO with N+1 pages of memory always performs as well or better than FIFO with N pages of memory.

False, see Belady's anomaly.

99) **LRU-K and 2Q** use both how recently and how frequently a page has been accessed to determine which page should be replaced.

True.

100) The **clock** policy replaces the least-recently-used page belonging to any process in the system.

False; clock approximates LRU, but it doesn’t necessarily replace the single least-recently-used page.

Part 2: Multiple-Choice Questions [4 points each]

Assume three jobs arrive at approximately the same time, but Job A arrives slightly before Job B, and Job B arrives slightly before job C. Job A requires 2 sec of CPU, Job B is 8 secs, and Job C is 7 secs. Assume a time-slice of 1 sec.

101) Given a FIFO scheduler, what is the **turnaround time** of job B?

a. 0 seconds
b. 2 seconds
c. 8 seconds
d. 10 seconds
e. None of the above

FIFO: A (until 2), B (until 10), C (until 17); B completes at time 10 seconds.

102) Given a FIFO scheduler, what is the **average response time** of the three jobs?

a. 1 second
b. 2 seconds
c. 4 seconds
d. 9.67 seconds
e. None of the above

Average response time: (0 + 2 + 10) / 3 = 4 seconds

103) Given a RR scheduler, what is the **turnaround time** of job B?

a. 1 second
b. 4 seconds
c. 16 seconds
d. 17 seconds
e. None of the above

B has the longest CPU burst and with RR will finish after every other job finishes, which is 2 + 8 + 7 = 17 seconds.
RR schedule: ABCABCBCBCBCBCBCB

104) Given a RR scheduler, what is the **average response time** of the three jobs?

a. 1 second
b. 2 seconds
c. 3 seconds
d. 12.33 seconds
e. None of the above

Average response time: (0 + 1 + 2) / 3 = 1 second.

105) Given a SJF scheduler, what is the **turnaround time** of job B?

a. 2 seconds
b. 9 seconds
c. 16 seconds
d. 17 seconds
e. None of the above

B is longest, so it is scheduled last; finishes when workload ends: 2 + 7 + 8 = 17.

106) Given a SJF scheduler, what is the **average response time** of the three jobs?

a. 2 seconds
b. 3.67 seconds
c. 9 seconds
d. 9.33 seconds
e. None of the above

Average response time: (0 + 2 + 9) / 3 = 3.67

Assume the OS schedules a workload containing three jobs with the following characteristics:

Job	**Arrival Time**	**CPU burst**
A	0	10
B	5	8
C	12	2

107) Which scheduler minimizes the completion time of the entire workload?

a. FIFO
b. RR
c. SJF
d. STCF
e. None of the above

All schedulers give the same result; the completion time for the entire workload (i.e., the last job) will be the same independent of the completion time of the individual jobs.

108) Given a STCF scheduler, what is the **response time** for job B?

a. 5 seconds
b. 7 seconds
c. 10 seconds
d. 20 seconds
e. None of the above

B is scheduled at time 10 (at time 5 when B arrives, it has a burst of 8 > A's remaining burst of 5); 10 -5 = 5.

109) Given a STCF scheduler, what is the **response time** for job C?

a. 0 seconds
b. 2 seconds
c. 12 seconds
d. 14 seconds
e. None of the above

When C arrives at time 12, it has a CPU burst of 2 < B's remaining CPU burst of 8-2=6; therefore, C is scheduled immediately when it arrives.

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 **0x0000037d (decimal 893)** and the bounds register contains **506 (decimal)**. Translate each of the following virtual addresses into physical addresses.

110) Virtual address 0x02e7 (decimal: 743) is physical address:

a. 0x000025d0 (decimal: 9680)
b. 0x00002581 (decimal: 9601)
c. 0x00003bc7 (decimal: 15303)
d. Segmentation Violation
e. None of the above

743 > bounds of 506

111) Virtual address 0x01ef (decimal: 495) is physical address:

a. 0x0000056c (decimal: 1388)
b. 0x00000e93 (decimal: 3731)
c. 0x00000d54 (decimal: 3412)
d. Segmentation Violation
e. None of the above

495 + 893 = 1388

112) Virtual address: 0x01a0 (decimal: 416) is physical address:

a. 0x00000c05 (decimal: 3077)
b. 0x0000051d (decimal: 1309)
c. 0x0000051e (decimal: 1310)
d. Segmentation Violation
e. None of the above

416 + 893 = 1309

Assume dynamic relocation is performed with a **linear page table**. Assume the address space size is 16KB, phys mem size is 64KB, and page size is 256 bytes. In a PTE, the high-order bit is the VALID bit. If the bit is 1, the rest of the entry is the PFN. If the bit is 0, the page is not valid. The following are the contents of the page table (from entry 0 down to the max size):

0x80000007
0x80000051
0x00000000
0x00000000
0x800000a1
0x800000d1
0x00000000
0x80000018
0x80000075
0x800000c2
0x8000000f
0x8000002c
0x80000015
0x8000004b
0x800000f1
0x8000006d
0x800000b0
0x800000d8
0x80000041
0x800000a5
0x800000ac
0x8000006f
0x8000002a
0x800000c7
0x800000e6
0x80000073
0x00000000
0x00000000
0x00000000
0x00000000
0x800000ea
0x800000e9
0x800000d0
0x800000f8
0x00000000
0x80000054
0x800000ce
0x800000c5
0x80000060
0x800000ac
0x80000087
0x800000b6
0x800000c6
0x00000000
0x8000004d
0x80000052
0x80000041
0x80000038
0x8000004e
0x80000055
0x800000e3
0x00000000
0x00000000
0x80000007
0x8000000d
0x8000006a
0x80000040
0x80000039
0x800000af
0x00000000
0x00000000
0x80000065
0x80000093
0x800000c0

113) Virtual Address 0x175e is physical address:
- a. 0x775e
- **b. 0xc75e**
- c. 0xd85e
- d. Invalid
- e. None of the above

Page size = 256 bytes → 8 bits for offset
0x1753 -> page 0x17 (decimal 16+7=23)
Contents of PTE 23 = 0x800000c7 (valid)
PPN: c7
Physical address 0xc75e

114) Virtual address 0x1940 is physical address:
- **a. 0x7340**
- b. 0xa540
- c. 0x51940
- d. Invalid
- e. None of the above

0x1940 → page 0x19 (decimal 16 + 9 = 25)
Contents of PTE 25: 0x80000073 (valid)
PPN: 73
Physical address: 0x7340

115) Virtual address 0x3b1e is physical address:
- a. 0x7b1e
- b. 0x0ble
- c. 0x00le
- **d. Invalid**
- e. None of the above

0x3b1e → page 0x3b (decimal 3*16 + 11 = 59)
Contents of PTE 49: 0x00000000 (valid bit not set!!!)

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
```

116) When accessing virtual address 0x3457, what will be the first page accessed (decimal)?

   a. 3
   b. 51
   c. 54
   d. 64
   e. Error or None of the above

Must first read page directory which we were told is stored in page 51 (pdbr = 51).

117) What contents (i.e., value) will be read on the first access (hexademical)?

   a. 0x0d
   b. 0x7f
   c. 0x92
   d. 0xaf
   e. Error or None of the above

Address 0x3457 in binary is: 01101 00010 10111.
With 32 byte pages, least-significant 5 bits (bit0-bit4) are used for page offset.
Bit10-bit14 are used for index into page directory: 01101 which is 8+4+1=13.
The 13th entry of page 51 is 0x92.

118) What will be the second page accessed (hex)?

   a. 0x0a
   b. 0x0c
   c. 0x1e
   d. 0x92
   e. Error or None of the above

0x92 in binary is 10010010. MSB of 1 means entry is valid. Rest of entry is 0x12. Next page table is stored on page 0x12, which is not one of the options...

119) What is the corresponding final physical address?

   a. 0x197
   b. 0x457
   c. 0x477
   d. 0x497
   e. Error or None of the above

We access page 0x12 (decimal 18) and look at entry 00010 (middle 5 bits of virtual address) and get the contents 0x8c. 0x8c in binary is 10001100; MSB of 1 means entry is valid. Rest of entry is 0x0c, which is our PPN. Remember the offset is 10111. Therefore, final address is 01100 10111, which in hex is: 0x197.

120) What contents (i.e., value) will be read from the final physical address?

   a. 0x05
   b. 0x07
   c. 0x1c
   d. 0x1d
   e. Error or None of the above

Go to physical page 0x0c (decimal 12) and look at offset 10111 (16+4+2+1 = 23). Contents are 0x1c.

121) When accessing Virtual Address 0x5830 what will be the first page accessed (decimal)?

   a. 51
   b. 54
   c. 58
   d. 59
   e. Error or None of the above

Again, for any virtual address in this address space, we always start with the page directory, which is stored in page 51.

Assume the OS is performing page replacement on only 4 pages of physical memory. Assume the following access stream of virtual pages:

Access: 8
Access: 7
Access: 4
Access: 2
Access: 5
Access: 4
Access: 7
Access: 3
Access: 4
Access: 5
Access: 9
Access: 5
Access: 2
Access: 7
Access: 6
Access: 2
Access: 9
Access: 9

122) If the OS uses the OPT replacement policy, how many misses will it incur?

- a. 8
- **b. 9**
- c. 10
- d. 11
- e. None of the above

Contents of 4 pages of physical memory:
8 - miss
87 - miss
874 - miss
8742 - miss
5742 - miss
5742
5742
5342 - miss
5342
5942 - miss
5942
5942
7942 - miss
7962 - miss
7962
7962
7962

123) If the OS uses the FIFO replacement policy, how many misses will it incur?

- a. 8
- b. 9
- **c. 10**
- d. 11
- e. None of the above

Access: 8: 8 miss
Access: 7: 87 miss
Access: 4: 874 miss

Access: 2 : 8742 miss
Access: 5 : 5742 miss
Access: 4 : 5742
Access: 7 : 5742
Access: 3 : 5342 miss
Access: 4 : 5342
Access: 5 : 5342
Access: 9 : 5392 miss
Access: 5 : 5392
Access: 2 : 5392
Access: 7 : 5397: miss
Access: 6 : 6397 miss
Access: 2 : 6297 miss
Access: 9 : 6297
Access: 9 : 6297

f.

124) If the OS uses the LRU replacement policy, how many misses will it incur?

a. 8
b. 9
c. 10
d. 11
e. None of the above

Access: 8 : 8 miss
Access: 7 : 87 miss
Access: 4: 874 miss
Access: 2: 8742 miss
Access: 5 5742 miss
Access: 4 5742
Access: 7 5742
Access: 3 5743 miss
Access: 4 5743
Access: 5 5743
Access: 9 5943 miss
Access: 5 5943
Access: 2 5942 miss
Access: 7 5972 miss
Access: 6 5672 miss
Access: 2 5672
Access: 9 9672 miss
Access: 9 9672

Congratulations on finishing a very long and detailed exam…