Memory management

(1) Show the typical memory hierarchy of a computer system. Explain the tradeoffs between latency (access times), capacity, and persistence, across different levels of memory hierarchy.

(2) What is a page table?

(3) How many page tables are maintained by the operating system?

(4) If there were no TLB, how would memory accesses be affected?

(5) What are the following? What do they do? Where are they located?

A. Memory Management Unit (MMU)
B. Translation Lookaside Buffer (TLB)
C. Page tables
D. Swap device

(6) Which statements that are true

A. A page table is an array of physical memory pages
B. Page table entries dictates whether a process can read, write, or execute the contents of a physical page.
C. A page table maps physical page numbers to virtual page numbers
D. Page table entries can be used to track which memory pages are infrequently accessed.

(7) How is a virtual address converted to a physical address in a virtual memory system? Explain the roles of MMU, TLB, and Page Tables.

(8) If you increase or decrease the page size in a system, how (and why) will it affect ***(a)*** the size of the page tables, and ***(b)*** the TLB miss ratio?

(9) What's TLB Coverage? Why is TLB coverage important?

(10) How can one increase the TLB coverage?

(11) In memory management, what is meant by relocation and protection? Why are they needed?

(12) A machine has a 32-bit address space and an 8-KB page. The page table is entirely in hardware, with one 32-bit word per entry. When a process starts, the page table is copied to the hardware from memory, at the rate of one word every 100 nsec. If each process runs for 100 msec (including the time to load the page table), what fraction of the CPU time is devoted to loading the page tables? (Assume that each process uses its entire virtual address space during execution.)

(13) A machine has a 32-bit address space and an 4KB page. The page table is entirely in hardware. Each page-table entry is 4 bytes in size. When a process starts, the page table is copied to the hardware from memory, at the rate of one byte every 25 nano-second. If each

process has a CPU burst of 200 msec (including the time to load the page table), what fraction of the CPU time is devoted to loading the page tables? (Assume that each process uses its entire virtual address space during execution.)

(14) Consider a machine having a 32-bit virtual address and 8KB page size.
   A. What is the size (in bytes) of the virtual address space of a process?
   B. How many bits in the 32-bit virtual address represent the byte offset into a page?
   C. How many bits in the 32-bit address are needed to determine the page number ?
   D. How many page-table entries does a process' page-table contain?

(15) Consider a machine having a 64-bit virtual address and 32KB page size.
   A. What is the size (in bytes) of the virtual address space of a process?
   B. How many bits in the virtual address represent the byte offset into a page?
   C. How many bits in the virtual address are needed to determine the page number ?
   D. How many page-table entries does a process' page-table contain?

(16) Consider a machine having a 64-bit virtual address and 16KB page size.
   E. What is the size (in bytes) of the virtual address space of a process?
   F. How many bits in the virtual address represent the byte offset into a page?
   G. How many bits in the virtual address are needed to determine the page number ?
   H. How many page-table entries does a process' page-table contain?

(17) For each of the following decimal virtual addresses, compute the virtual page number and offset for a 4-KB page, an 8 KB page, and a 16KB page: 20000, 32768, 60000.

(18) A computer with a 32-bit address uses a two-level page table. Virtual addresses are split into a 9-bit top-level page table field, an 10-bit second-level page table field, and an offset. How large are the pages and how many pages are there in the address space?

(19) Which system components handle TLB misses and page-faults, and how, in a machine with
   (a) architected page-table?
   (b) architected TLB?

(20) What is the purpose of "Referenced" and "Modified" ("Dirty") bits in the page table entry? How are they manipulated by the (a) hardware, and (b) operating system?

(21) Consider a virtual memory system running on an architected page-table hardware supporting two-level page tables. Page tables are not locked in memory and may be swapped to disk. An lw (load word) instruction reads one data word from memory; the address is the sum of the value in a register and an immediate constant stored in the instruction itself. Neither machine instructions nor page-table entries nor data words can cross a page boundary. In the worst case, how many page faults could be generated as a result of the fetch, decode, and execution of an lw instruction? Explain why?

(22) What is meant by "Internal" fragmentation?

(23) What is "External" fragmentation of memory? How can it be resolved?

(24) Consider two processes that set up one page of shared memory for inter-process communication with each other. Given what you know about virtual memory management, explain how the OS would set up this shared memory page at the level of page tables?

(25) Suppose that the Operating System wanted to track (or intercept) every write performed to a specific memory page by a user-level process. Explain how the OS would achieve this goal?

(26) How can the operating system track
   A. Dirty (or updated) memory pages for the purpose of eviction?
   B. Every memory write performed by a process to specific memory pages?

(27) Considering memory protection, explain how the operating system ensures that user-level processes don't access kernel-level memory?

(28) Suppose that "page table pages are paged", meaning that (some or all of) the memory allocated to hold page tables can be paged-in and out of the main memory by the operating system. Suppose further that you have two-level page-tables, i.e. a first-level page-directory which tracks the second-level page table blocks.
   A. Which parts of the page-table can be paged (moved in and out of main memory)?
   B. Where are the memory address translations (i.e. page table entries) for the "paged page-table"?
   C. Can the memory used for your answer in (b) be paged? Why? Or Why not?

Superpages

1. What are superpages? Why are they useful? What are the constraints on their sizes and placement?
2. Superpages
    a. What are the advantages and disadvantages of superpages?
    b. How many TLB entries and page table entries are there for each superpage?
    c. What are the restrictions on superpage size, allocation, and placement?

3. If a superpage has a size equal to 4 base pages, how many TLB entries are occupied by the superpage? How many page table entries are occupied by the same superpage? Explain why.

4. How do superpages affect (increase/decrease/don't change) internal and external memory fragmentation? Why?
5. In the paper, superpage allocations are performed in two steps: preemptible reservations and incremental promotions. Why do we need this two-step mechanism? If we already know how much to reserve, why not create the entire superpage in one shot the first time any of its base pages is accessed?
6. Why is it that using a mix of multiple superpage sizes tends to yield better application speedups than using superpages of only one size?
7. In the superpage paper, explain how ***preemptible reservations***, ***incremental promotions*** and ***speculative demotions*** work.
8. Which fragmentation (Internal/external) is made worse by superpages? Why?

9. How does TLB Coverage and TLB miss ratio vary with the size of a page?

10. In the superpage paper, how does the system proposed avoid the need to page-out an entire superpage to the disk upon memory pressure? What is the source of performance overhead in this mechanism?

11. Use of superpages (as they are currently designed) requires the use of contiguity restoration techniques in the operating systems. To avoid the need for contiguity restoration, one could get rid of the requirement that the base physical pages of a superpage be contiguous - in other words, allow the base pages in physical memory not to be next to each other. If one does so, describe how you would redesign (if at all) the TLB, page-tables, and the mechanism for translating virtual to physical addresses? You can assume either architected page-tables or architected TLB. Remember to list any assumption you make, justifying why they are reasonable.

12. In the "superpages paper" (Navaro et. al.), when and how are (a) reservation, (b) promotion, and (c) demotions carried out? (d) Why are reservations called "pre-emptible"?

13. Which of the following memory designs can cause internal fragmentation only, external fragmentation only, both, or neither? Briefly explain why.
    A. Pure paging
    B. Pure segmentation
    C. Paging with Segmentation

D. Using superpages of the same size (no base pages or any other superpage size)
E. Using a mix of superpages of different sizes

Segmentation

1. How many page tables are there per segment in Multics and Pentium. (b) Which one (Multics/Pentium) is better? Why?

2. What problem does segmentation solve that paging doesn't solve? What problem does paging solve that segmentation doesn't solve?

3. How are relocation and protection implemented in Pentium architecture? Consider the roles of both segmentation and paging.
4. How is segmentation different from paging? Why was each technique invented?
5. Using either Multics or Pentium architecture as an example, explain how segmentation is used in enforcing protection?
6. How is a virtual address converted to a physical address, considering both segmentation and paging in (a) Multics and (b) Pentium architectures?

Virtualization

1. For system virtual machines, explain how virtual memory addresses are translated to physical addresses when (a) hardware supports EPT/NPT (extended/nested page tables) and (b) hardware only supports traditional (non-nested) page tables.

2. How does Intel VTx extend the traditional CPU execution privilege levels to support system virtual machines?

3. Compare different approaches for virtualizing I/O devices for virtual machines.

4. Explain briefly with examples (1) Process virtual machine, (2) System virtual machine, (3) Emulator, (4) Binary optimizer, (5) High-level Language Virtual Machine.

5. Which interface does a Process VM virtualize? Which interface does a System VM virtualize?

6. (a) How do Interpreters differ from Dynamic Binary Translators? (b) How do Binary Optimizers differ from Emulators?

7. What are the advantages and disadvantages of Classical System VMs compared to Para-virtualized VMs?

8. Give at least three mechanism(s) by which the highest privileged software, such as an operating system or a hypervisor, retains control over the CPU?

9. What is a co-designed virtual machine? Briefly describe and give an example.

10. What type of virtual machine (VM) is each of the following and why? Be as specific as possible. (a) Java Virtual Machine (JVM) (b) VMWare (c) Xen (d) Digital FX!32 (e) VirtualPC (f) (e) Transmeta Crusoe (Code Morphing)

11. Explain the difference between the concepts of full-virtualization and para-virtualization, giving at least one example of both virtualization techniques.

12. Let's say that you are asked to modify the Linux OS so that programs and libraries compiled on Windows OS could run natively on Linux, meaning they should be executed as normal programs (without using any emulator or virtual machine). What would be your high-level approach?

13. (a) What is a shadow page table? How is it constructed and/or updated by the hypervisor? (b) What type of hardware virtualization support is needed to avoid constructing shadow page-tables in full-virtualization?

OS-level virtualization

1. What is OS-level virtualization? OS-level as opposed to what other levels?
2. How would you define the notion of isolation for any software or system component? What are the two (or more) extremes?
3. (a) What are “containers”? (b) What type of virtualization do they provide? Explain.
4. How does the isolation model of containers differ from traditional system VMs and process VMs?
5. How does the isolation model for a process differ from that for a system virtual machine?
6. How do containers improve upon process-level isolation model?
7. What are jails or sandboxes in the OS world? Why would you use them?
8. What are some key system resources that are virtualized by jails/containers/sandboxes?
9. Linux containers contain consist of two main components: Namespaces and CGroups. Explain the role of each.

Security

1. What is the difference between security and privacy? Are they entirely the same? Or entirely different?
2. Explain the three key principles of computer security?
3. What is a threat model? What factors should you consider when defining threat model?
4. What hardware mechanism does x86 ISA provide to ensure that Operating System's code and data are protected from user-level processes?
5. What is the role of privilege levels (defined by the ISA) in a computer system? How many privilege levels are defined in the x86 ISA? In which privilege level does the OS execute?
6. Explain the basic security mechanisms supported by (a) the CPU execution hardware, (b) Memory management hardware and software, (c) File system. Assume that the machine uses x86 ISA.

7. In x86, how does the MMU figure out whether a code currently executing on CPU has permissions to read/write to/execute a given address in memory?

8. What is authentication?
9. Describe different techniques to authenticate users.
10. What are some ways in which by which authentication mechanisms can be subverted?
11. What's a computer virus? What's a computer worm?
12. Explain a buffer overflow attack.
13. What is sandboxing? List two sandboxing mechanisms.
14. Explain Discretionary, Mandatory, and Role-based access control mechanisms.
15. Explain (a) trusted computing base (TCB) including why is it called "Trusted", (b) Reference Monitor, and (c) relationship between TCB and reference monitor.

16. Explain the two key data access principles of multi-level security (MLS) systems (also called Mandatory Access Control).

17. Why is Mandatory access control called "mandatory"? What's the alternative?

18. What type of systems require mandatory access control?

19. Give an example of a scenario where the software doesn't trust the OS, hypervisor, and/or the hardware platform on which it runs? What can the software possibly do to "secure" itself in this situation?

20. Considering memory protection, explain how the operating system ensures that user-level processes don't access kernel-level memory?