# Virtual Memory

**(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) What is a page fault and a TLB miss? Which system component resolves each of them?

(7) Mark the 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.

(9) How is a virtual address converted to a physical address (considering only paging) ? Explain the roles of MMU, TLB, and Page Tables.

(10) Why does a memory access violation in the kernel result in a kernel crash?

(11) 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?

(12) What is TLB coverage and why is it important?

(13)What are two ways to increase TLB coverage?

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

(19)Consider a machine with B-bit architecture (i.e. virtual address and physical address are B bits long). Size of a page is P bytes.

A. What is the size (in bytes) of the virtual address space of a process?
B. How many bits in an address represent the byte offset into a page?

C. How many bits in an address are needed to determine the page number ?
D. How many page-table entries does a process' page-table contain?

(20) 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.)

(21) 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.)

(22) 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?

(23) 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?

(24) Consider a machine having a 64-bit virtual address and 16KB 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?

(25) 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?

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

(b) architected TLB?

(27) 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?

(28) What is the difference between internal and external fragmentation?

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

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

(31) What is a "working set"? Why is it so important?

(32) 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?

(33) 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?

(34) 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?

(35) How does TLB Coverage and TLB miss ratio vary with the size of a page?

(36) 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?

(37) 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?