a. Page table - where the operating system stores its mappings of virtual addresses to physical addresses

b. TLB - a cache exclusively for the page table which allows the page table to look up the physical addresses at a cache level speed

c. Multi-level page table – page table of page tables, first table is broad linking to other tables that cover all available memory.

d. Memory mapping a file - a file, we can greatly increase I/O performance, especially on large files. It is however possible that by memory-mapping a file, we create more page faults, which could actually slow down performance.

e. Demand paging - A disk page only gets copied into physical memory if an attempt is made to access it. This can cause a very large number of page faults if memory is rarely re-used, but will save a lot of disk lookups if memory is reused constantly.

2. How does virtual memory simplify memory management for the compiler?

   Virtual memory allows the compiler to always start at memory address 0, and makes finding "available" chunks of contiguous memory very easy, since it doesn't have to be continuous in physical memory.

3. How does virtual memory protect address spaces from each other?

   By using paged virtual memory, it makes it impossible for a program to access a page that is not allocated to that program. Attempting to do so would cause a page fault.

4. How does the OS handle a page fault?

   The operating system tries to handle the page fault by making the required page accessible at a location in physical memory or kills the program in the case of an illegal access.

5. How does virtual memory allow multiple programs to share the same memory?

   When using virtual memory we can have multiple virtual addresses that all point to the same chunk of virtual memory (read-only access) allowing the memory to be shared.

6. How do multi-level page tables reduce the overhead due to virtual memory?

   A normal page table has to have an entry for every single memory address, whether its being used or not. A multi-level page table is essentially a page table of page tables. The first page table has very broad links to other page tables that cover all available memory. The second layer of page tables only have entries for addresses that are actually used. For your average program that doesn't use all or most of the available memory, this causes a large speedup. However if a program is using all available memory and thrashing then this approach will cause an even larger drop in performance.

7. What does the valid bit indicate in a TLB entry?

   The valid bit of a TLB says whther the entry is currently usable. If the valid bit is 0, the computer must go out to disk to find the page.

8. How does memory-mapped file I/O differ from normal I/O from a programmer's perspective?

   CPU instructions that can access memory can also be used for accessing devices because memory-mapping uses the same address bus.

9. Give an example where demand paging could be less efficient than normal I/O?

   If you will be using most or all of the pages, then demand paging will cause several page fault traps, which cause a program to slow down immensely

.

10. How does memory-mapping speed typical the Fork() operation?

    Memory mapping removes the need to copy the pages of the original parent by allowing the child to share the pages.