TUTORIAL SHEET - WEEK OF OCTOBER 17th

1. What is the function of a TLB? What is its structure? How many entries does it have?
2. Given a 32 bit addressing mode, a 4KB page size, how many entries does a two level paging scheme that uses 10 bits for the primary page table have.
3. Assume that a machine has 1 GB of main memory, what is the largest amount of swap space required, assuming that we never maintain two copies of a frame?
4. How can multiple processes share data with the help of the paging mechanism?
5. How should we maintain the notion of precise exceptions in the presence of page faults (misses in the page table)?
6. Assume a process A, forks a process B. *Forking* a process means that B will inherit a copy of A's entire address space. However, after the fork call, the address spaces are separate. How can we implement this using our paging mechanism?
7. Programs tend to invoke tasks in the operating system very often. This means that we need to unload and reload page tables very often. How can we eliminate this effect by making changes to the compiler and the paging mechanism.
8. We have a producer-consumer interaction between processes A and B. A writes something that B reads in a shared space. However, B should never be allowed to write anything into that shared space. How can we implement this using paging? How do we ensure that B will never be able to write into the shared space?
9. Instead of having a separate page table per process, we can have an inverted page table that is indexed by the frame number. Each cell would contain the page number. How should we go about designing this?
10. How do dynamically linked libraries use the features provided by virtual memory?