Limited Direct Execution (Ch 6.) Discussion Questions

- What does architecture mean?
    - How does it differ from OS?
- What is the key question/problem addressed in this chapter?
    - Why not just run processes directly on the CPU & let them yield when done?
    - Why not have the OS fetch, decode, & execute each instruction of a program instead?
- What is an interrupt?
- **Restricted Operations**
- Why are there restricted operations for processes?
- What is user mode? kernel mode?
    - what type of operations are restricted in user-mode?
    - who/what provides these two modes?
- How does a process get things done that require restricted operations?
- What does a system call do?
- Who/what provides the trap/return for trap instructions?
- How does the CPU know which code to run when a trap occurs?
- **Context Switching**
- How does direct execution make it difficult for the OS to switch between processes?
- How does the OS solve this problem?
- What is the difference between the cooperative and non-cooperative approach to context switching?
    - How do they work?
    - What are the drawbacks of cooperative?
    - " " non-cooperative?

    - how does the CPU know what code to run when the timer interrupt occurs?
- What does the scheduler do (in the simplest terms)?
- How does limited direct execution allow the OS to virtualize the CPU?