**CS 383 Final  
Katie Prescott**

**12/14/17**

1. When a system has multiple levels of cache memory, L2 always has more memory than L1. Why is this necessary?
   1. **L2 exists to speed up the case where there is an L1 cache miss. If L2 wasn’t bigger than L1, it would not be able to have more lines than L1 to deal with L1 cache misses.**
2. Explain how pipelining serves to reduce the average number of steps in the execution part of the fetch-execute cycle.
   1. **When pipelining is used, the CPU and ALU can be designed faster and more complex. Pipelining increases the performance, it doesn’t reduce the amount of time to complete an instruction, but increases the amount of instructions being processed at once, making the program run faster.**
3. In general, what purpose does an interrupt serve? Stated another way, suppose there were no interrupts provided in a computer. What capabilities would be lost?
   1. **Interrupts increase the overall efficiency of a computer. Without interrupts, the processor would need to poll every device periodically to check if any of them need attention.**
4. Consider the interrupt that occurs at the completion of a disk transfer.
   1. ‘‘Who’’ is interrupting ‘‘whom’’?
      1. **The disk controller sends interrupt to CPU**
   2. Why is the interrupt used in this case? What would be necessary if there were no interrupt capability on this computer?
      1. **Interrupt notifies CPU when disk transfer is finished so the CPU can use the necessary memory. Without interrupt capability, CPU would need to poll to determine if the transfer is complete.**
   3. Describe the steps that take place after the interrupt occurs.
      1. **CPU services interrupt, and notifies the program that the disk transfer can continue. The CPU continues executing from the point after the transfer request.**
5. Describe the steps that occur when a system receives multiple interrupts.
   1. **When multiple interrupts occur, they get put into a priority queue of interrupts. The programs are then executed based on the priorities of the interrupt.**
6. What is the difference between polling and polled interrupt processing?
   1. **Polling: An interrupt signal that includes which device is sending the interrupt  signal.**
   2. **Polled interrupt: An I/O interrupt that notifies the computer that a device is ready  to be read/handled, but doesn’t include which device. It would need to poll each  device to find which one needs attention.**
7. Explain the major differences between TCP and UDP. Explain how an Ethernet frame provides synchronization between sender and receiver nodes.
   1. **TCP: computers connect via sockets before data gets sent. Data gets sent in numbered packets and require acknowledgement messages from receiver to sender to verify receipt. If packet is not acknowledged, TCP resends it.**
   2. **UDP: there is not connection set up before data gets sent. Data cannot be verified. UDP packets are sent separately, it is simpler and faster, but is not as reliable as TCP.**
8. Identify and briefly explain the four fundamental topologies used in networks.
   1. **Mesh: Multiple paths between end nodes. If an individual intermediate node fails, network traffic slows, but does not stop if there is an alternative path available.**
   2. **Bus: All nodes connected to single line bus. Messages sent along wire and received by all nodes, but only read by node it’s addressed to.**
   3. **Star: All nodes get connected to, and communicate through, a central device.**
   4. **Ring: Nodes connect to the next node in a network, the last one is connected to the first. Packets sent through loop until they reach the node they are addressed to.**
9. Explain the Client – Server network architecture. What the difference with a Peer-to-Peer network?
   1. **The client-server model is a distributed application structure. It partitions tasks between the servers (provide a service) and clients (request a service). The peer-to-peer network is when 2 or more computers are connected and share resources without a server.**
10. Explain *dispatching*. Describe the two basic methods that are used by operating systems to implement dispatching.
    1. **A dispatcher (short term scheduler) gives the CPU control to the process chosen by the scheduler. A switch to user mode and jumping to the right location is the program is required. When a process is interrupted, the process gets put into a queue, and the dispatcher selects which process from the queue to execute.**
11. Explain the concept of a *process*. How does a process differ from a program? Explain *concurrent processing*. Briefly describe at least two services that an operating system must provide to support concurrent processing.
    1. **A process is a program that is currently being executed, along with the allocated resources. A program is just a list of steps in memory. Concurrent processing is when multiple processes are running at the same time. The OS must allocate memory and resources for each program. CPU time must also be fairly given for each program.**
12. What is the fundamental purpose of any operating system? What is the role of the file manager? What other basic functions must the operating system be able to perform?
    1. **The operating system interacts with the hardware, interpreting programs. It is the platform for all other applications on a computer. The file manager provides an interface for the setup of the file system, including navigation, access and manipulation commands, and a way to read and edit the files. The operating system also needs to provide interfaces for the user and their programs, support for I/O services, means for initial startup, interrupt processing, networking services, security, and provide information and tools to the user.**
13. What features characterize threads? How are threads used? Explain *deadlock*. What are the three possible ways that an operating system can handle the issue of deadlock?
    1. **A thread is the smallest sequence of instructions that can be managed on its own by a scheduler. Deadlock is when every process is blocked, meaning there is some shared resource that cannot be accessed by the other processes, but needs other resources that are held by other processes. An operating system can handle deadlock by preventing it from ever occurring (using specific protocols, like semaphores), it can allow a deadlock to occur, then delete and recover from it, or it can ignore the problem, pretend it doesn’t exist.**