COSC1112/1114: Operating Systems Principles

Tutorial 10 (week 11)

1. **When multiple interrupts from different devices appear at about the same time, a priority scheme could be used to determine the order in which the interrupts would be serviced. Discuss what issues need to be considered in assigning priorities to different interrupts.**

Answer:
A number of issues need to be considered in order to determine the priority scheme to be used to determine the order in which the interrupts need to be serviced. First, interrupts raised by devices should be given higher priority than traps generated by the user program; a device interrupt can therefore interrupt code used for handling system calls. Second, interrupts that control devices might be given higher priority than interrupts that simply perform tasks such as copying data served up a device to user/kernel buffers, since such tasks can always be delayed. Third, devices that have real-time constraints on when their data is handled should be given higher priority than other devices. Also, devices that do not have any form of buffering for its data would have to be assigned higher priority since the data could be available only for a short period of time.

2. **What are the advantages and disadvantages of supporting memory mapped I/O to device-control registers?**

Answer:
The advantage of supporting memory-mapped I/O to device-control registers is that it eliminates the need for special I/O instructions from the instruction set and therefore also does not require the enforcement of protection rules that prevent user programs from executing these I/O instructions. The disadvantage is that the resulting flexibility needs to be handled with care; the memory translation units need to ensure that the memory addresses associated with the device control registers are not accessible by user programs in order to ensure protection.

3. **Some DMA controllers support direct virtual memory access, where the targets of I/O operations are specified as virtual addresses and a translation from virtual to physical address is performed during the DMA. How does this design complicate the design of the DMA controller? What are the advantages of providing such a functionality?**

Answer:
Direct virtual memory access allows a device to perform a transfer from two memory-mapped devices without the intervention of the CPU or the use of main memory as a staging ground; the device simply issues memory operations to the memory-mapped addresses of a target device and the ensuing virtual address translation guarantees that the data is transferred to the appropriate device. This functionality, however, comes at the cost of having to support virtual address translation on addresses accessed by a DMA controller and requires the addition of an address-translation unit to the DMA controller. The address translation results in both hardware and software costs and might also result in coherence problems between the data structures maintained by the CPU for address translation and corresponding structures used by the DMA controller. These coherence issues would also need to be dealt with and results in further increase in system complexity.

4. **Typically, at the completion of a device I/O, a single interrupt is raised and appropriately handled by the host processor. In certain settings, however, the code that is to be executed at the completion of the I/O can be broken into two separate pieces. The first piece executes immediately after the I/O completes and schedules a second**

interrupt for the remaining piece of code to be executed at a later time. What is the purpose of using this strategy in the design of interrupt handlers?

Answer:

The purpose of this strategy is to ensure that the most critical aspect of the interrupt handling code is performed first and the less critical portions of the code are delayed for the future. For instance, when a device finishes an I/O operation, the device-control operations corresponding to declaring the device as no longer being busy are more important in order to issue future operations. However, the task of copying the data provided by the device to the appropriate user or kernel memory regions can be delayed for a future point when the CPU is idle. In such a scenario, a lower-priority interrupt handler is used to perform the copy operation.